Universistat de Lleida
Escola Politècnica Superior
Enginyeria Tècnica en Informàtica de Gestió

Trabajo de final de carrera

Estudio de los futuros estándares HTML5 y CSS3. Propuesta de actualización del sitio www.mpiua.net

Autor: Josep Mir Huguet
Director: Dr. Toni Granollers i Saltiveri
Septembre 2012

Universistat de Lleida

Escola Politècnica Superior

Enginyeria Tècnica en Informàtica de Gestió

Trabajo de final de carrera

Estudio de los futuros estándares HTML5 y CSS3. Propuesta de actualización del sitio www.mpiua.net

Autor: Josep Mir Huguet

Director: Dr. Toni Granollers i Saltiveri

Septembre 2012

AGRADECIMIENTOS

A mis padres,
ya que sin su apoyo incondicional y su gran esfuerzo realizado en todos los aspectos jamás hubiera tenido la oportunidad de cursar estudios superiores.

A Toni Granollers,
por acogerme para la realización de este proyecto, por su dedicación y su trato tan cercano y amigable que me ha brindado durante todo este tiempo.

A todos los profesores y compañeros,
que a lo largo de la titulación me he ido encontrando y que han hecho que el camino se hiciera mucho más placentero.

A todos mis amigos,
que me han dado ánimos constantemente para seguir adelante durante la realización de este proyecto.

A todos,
Muchas Gracias.

Tabla de contenido

CAPITULO 1: INTRODUCCION

1.1. Introducción/Resumen

Este trabajo de final de carrera se divide en dos partes, primeramente se ha realizado un estudio de los futuros estándares HTML5 y CSS3, y en la segunda parte se ha propuesto una actualización en el sitio web www.mpiua.net utilizando dichos estándares.

El estándar HTML5 nos proporciona una notoria mejora en la implementación de páginas web. Este nuevo estándar nos proporciona una nueva forma de estructurar el contenido, recordándonos, por su similitud, a la estructuración de un periódico. Contando con sus cabeceras, secciones, artículos, pies de página,…

Además de esto, este nuevo estándar pone a nuestra disposición una serie de APIs[1] que nos permiten desarrollar aplicaciones web de una manera mucho más cómoda y sencilla. Estas APIs no abarcan todas las implementaciones posibles, pero si los mas utilizados dentro del marco tecnológico actual.

Por otra parte, CSS3 nos proporciona una herramienta para controlar el diseño de nuestra web también muy acorde con el con el panorama actual de las webs dinámicas. Además nos proporciona una serie de efectos y animaciones totalmente nuevas y revolucionarios en el campo de las hojas de estilo.

Una vez conocidas las nuevas características y el alcance de esta nueva tecnología, se ha procedido a realizar una propuesta de actualización del sitio web www.mpiua.net, con el único objetivo de adaptar dicho sitio web a los nuevos estándares. La propuesta que se ha realizado es cambiar la estructuración de las paginas, así como la utilización de las nuevas APIs y del propio CSS3. A pesar de estos cambios realizados, se ha preservado al máximo el diseño original del sitio web pues no entra en los objetivos de este trabajo ni el estudio de su diseño ni de su arquitectura de la información.

1.2. Objetivos

El objetivo de este trabajo final de carrera es realizar un estudio de todas las nuevas características y funcionalidades que nos ofrece tanto HTML5 como CSS3. Con ello pretendemos entender que mejoras supone respecto a las versiones anteriores y ejemplarizarlo aplicándolas en un sitio web real concreto. Este sitio web será el portal de *Modelo de Proceso de la Ingeniería de la usabilidad y de la accesibilidad, MPIu+a.*

[1] Interfaz de Programación de Aplicaciones.
http://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones.

1.3. Motivación

La realización de un proyecto de estas características era algo que me recorría por dentro desde que empecé a cursar la asignatura “Introducción a la Interacción Persona-Ordenador” (abreviada por InIPO) en segundo curso de la titulación. A medida que profundizamos en los temas de accesibilidad y usabilidad me empecé a dar cuenta de las carencias que tenían muchísimos sitios web que visitaba a menudo.

Está mal implementado aquel portal web carezca de sistemas de accesibilidad, porque crea una barrera, a un conjunto de usuarios mucho más grande de lo que comúnmente se piensa. Pero que una página web no sea ni usable significa que no sirve absolutamente para nada, ya que no la va a utilizar nadie. Y desgraciadamente hay muchísimos lugares donde se dan estos casos.

También me interesaba hacer una mirada hacia el futuro, como son los estándares de edición web, especialmente HTML5 y CSS3, ya que creo que tendrán una gran repercusión en un futuro no muy lejano. Ya hay muchas aplicaciones de dispositivos móviles basados en esta tecnología, y en breve empezaran a comercializarse los primeros sistemas operativos, con lo cual creo que es un marco novedoso y digno de estudio como ampliación a la formación aprendida a lo largo de la titulación.

Veo interesante introducirme en este campo, puesto que el concepto de maquina local, aplicación local y datos locales esta cambiando y cada vez se orienta mas hacia la nube, con lo cual creo que seria un error no adentrarse y conocer un poco este entorno.

Por estas razones me dirigí al Dr. Toni Granollers para ver si podía dirigirme algún trabajo que englobara estas inquietudes y como puede verse han sido todas satisfechas.

1.4. Temporalización y estimación económica

En este punto vamos a exponer el tiempo empleado en la realización del trabajo y a intentar hacer una valoración económica.

Temporalización del proyecto.

La temporalización empleada es la que se puede apreciar en la figura1_01.

Id	Id	Actividad	Inicio	Fin
1		Acuerdo del proyecto	08/02/2012	08/02/2012
2		Negociación del proyecto	08/02/2012	22/02/2012
3		Aprendizaje de implementación de paginas web	28/02/2012	20/04/2012
4		Primera toma de contacto con HTML5 y CSS3	12/03/2012	14/06/2012
5		Negociación del proyecto II	19/06/2012	20/06/2012
6		Estudio HTML5	22/06/2012	07/08/2012
7		Documentación	26/06/2012	30/08/2010
8		Estudio CSS3	11/07/2012	21/07/2012
9		Implementación de la web.	08/06/2012	28/08/2012

Figura1_01: Tabla de temporización de actividades.

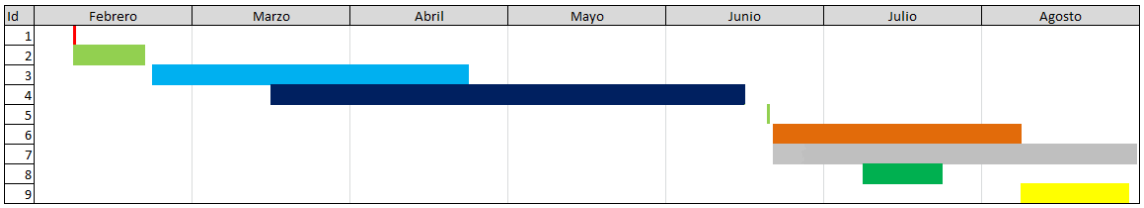

Figura1_02: Diagrama de Gantt de la temporalización del trabajo.

En la figura1_02 tenemos un diagrama de Gantt de las diferentes actividades realizadas.

Aunque en el diagrama de Gantt aparezcan solo dos situaciones de negociación el proyecto, decir que constantemente he estado en contacto con el director para resolver todas las dudas que me iban surgiendo. Lo he marcado de esta forma ya que eran unas sesiones más intensas, y las predecesoras a los dos grandes partes de temporalización de este proyecto, que son la preparación y la realización del mismo.

La primera actividad de *acuerdo del proyecto* corresponde al momento en que me pongo en contacto con el director del proyecto y acordamos que realizare el trabajo.

La segunda actividad corresponde al periodo en el cual estamos decidiendo el perfil del proyecto, el que y el como se hará.

La tercera actividad pertenece al periodo que corresponde al aprendizaje de la construcción e implementación de páginas web. Decir que esta tarea no se realiza con mucha dedicación puesto que estaba cursando otras asignaturas de la titulación. Es un trabajo bastante discontinuo.

La cuarta tarea es la iniciación al propio código HTML5 y CSS3, una primera toma de contacto y recoger información acerca de esta nueva tecnología, como son libros, tutoriales, hacer pruebas,… Al igual que la tarea anterior, esta tampoco se ha realizado de manera continuada por el mismo motivo.

La quinta tarea pertenece al segundo momento en que me reúno con el director del proyecto, y con el conocimiento aprendido acerca de HTML5 y CSS3 se pacta la forma definitiva que tendrá el proyecto.

La sexta tarea comprende el estudio de HTML5, decir que esta tarea ya esta medio planeada, puesto que en la cuarta tarea ya se ha recopilado toda o casi toda la información necesaria para realizar tal estudio.

La séptima tarea pertenece a la documentación del trabajo, que se empieza a realizar junto con el estudio y es la última tarea con la que acaba el trabajo.

La octava tarea pertenece al tiempo de estudio dedicado al estándar CSS3. Este estudio se realiza al mismo tiempo que el de HTML5, pero con una mayor dedicación.

La novena tarea pertenece a la implementación de la nueva web, este tiempo representa el tiempo empleado en implementar todo el portal web, empezando desde cero.

Valoración económica del trabajo.

La valoración del proyecto la realizaremos en dos partes por separado, una parte que será la implementación de la nueva web, y la otra parte será la realización del estudio de HTML5 y CSS3.

Para la implementación de la web se ha valorado el precio de la hora en 8 € (soy conocedor que se trata de un precio totalmente orientativo y que puede no ser real, pero ayuda a proporcionar una estimación económica). Durante la realización de esta tarea, se hacían 10 horas al día de media aproximadamente, y también se trabajo sábados y domingos, por lo tanto:

20 días x 10 horas x 8 € = 1600 €

Para calcular el presupuesto del tiempo dedicado al estudio de HTML5 y CSS3, hablamos de un precio por hora de 5 € (este precio es ligeramente inferior ya que considero que al no estar produciendo nada y además recibiendo una formación a cambio, no puede estar valorada de la misma manera). Durante el trascurso de esta tarea se realizaba una media de 7 horas al día y un día de descanso semanal. Por lo tanto de deducimos que:

46 días x 7 horas x 5€ = 1610 €

Por consiguiente, con los supuestos anteriores, el coste final del proyecto ascendería a 3.210€.

1.5. Estructura del documento.

Este documento contiene seis capítulos, cada uno de ellos contiene lo siguiente:

Capitulo1: INTRODUCCION Este primer capitulo contiene una introducción de lo que veremos en el trabajo. Se han especificado los objetivos de este, así como la motivación que ha habido para llevarlo a cabo. Contiene otro punto donde se realiza una explicación de la temporalización y su valoración económica, además de este punto donde se explica la estructuración del documento.

Capitulo2: HTML5 En este capitulo se ha procedido a la realización del estudio del estándar HTML5. Primeramente se realiza una breve introducción al HTML y a HTML5. Después se explica la nueva semántica que propone este estándar, seguida de los nuevos formularios y de una explicación de las APIs que incorpora HTML5.

Capitulo3: CSS3 Al igual que el capitulo anterior, también se realiza una pequeña introducción a las hojas de estilo y a CSS3. Y a lo largo del capitulo se ven las distintas novedades que este nuevo estándar nos proporciona.

Capitulo4: CASO PRÁCTICO. PROPUESTA DE ACTUALIZACION DEL SITIO WEB: MPIU+A – Modelo de Proceso de la Ingeniería de la Usabilidad y de la Accesibilidad En este capitulo se explican todas las novedades que se han implementado en la nueva pagina web.

Capitulo5: CONCLUSIONES Y TRABAJO FUTURO Este capitulo contiene las conclusiones personales y una reflexión a un trabajo futuro.

Capitulo6: BIBLIOGRAFIA Este es el capitulo final, y es donde se muestra todos aquellos recursos ya sean libros, paginas web, documentos o video-tutoriales, que se han empleado para realizar este trabajo.

CAPITULO 2: HTML5

2.1. Estándar HTML

El HTML, de las siglas en ingles HyperText Markup Language, es el lenguaje más utilizado para la elaboración de páginas web. Este lenguaje se utiliza para mostrar texto, acompañado de otros elementos como imágenes o tablas, a través de internet. Para poder visualizar estas páginas e interactuar con ellas el usuario necesita utilizar un "navegador web", programa informático que interpreta el lenguaje HTML y lo visualiza en la interfaz del dispositivo.

El lenguaje HTML es un lenguaje de texto, el cual podemos crear y editar con cualquier editor de texto que tengamos a nuestra disposición, como por ejemplo notepad, gedit... También existen editores específicos para HTML más completos y con más herramientas para este lenguaje, es el caso de DreamWeaver de Adobe, WebStorm de Jetbrains, WebMatrix de Microsoft seguido de un largo etcétera.

Es un lenguaje bastante sencillo que se basa en la utilización de etiquetas para definir a cada elemento del documento. Las etiquetas deben definirse al principio y al final de cada elemento. Estas etiquetas se encierran en las claves < y >; y para marcar la finalización de uso de una etiqueta utilizamos otra vez la misma etiqueta pero añadiendo / detrás de la clave de apertura tal que así:

```
<etiqueta>Contenido</etiqueta>
```

Si la etiqueta que queremos definir no contiene nada, podemos cerrarla utilizando / antes de la clave de cierre tal que así:

```
<etiqueta />
```

Este lenguaje se diseño en un principio para divulgar texto y alguna imagen acompañándolo, en algunos sitios; nunca se pensó que llegaría a ser tan utilizado y tener las necesidades que tiene hoy en día, por eso las primeras implementaciones eran tan pobres. Tras este hecho, todas las nuevas versiones que han aparecido, contenían mejoras no solo para los usuarios, si no también para los navegadores y desarrolladores de páginas web, pues estos también habían aumentado en número.

2.2. El estándar HTML: la nueva versión HTML5

HTML5 es la quinta revisión, que todavía esta en fase de desarrollo, del lenguaje por excelencia para elaborar páginas web. Esta nueva versión no solo contiene la nueva revisión del estándar

de HTML, si no que agrupa un conjunto de utilidades para desarrollar páginas y con ellas aplicaciones web con la ayuda de javascript[2].

En esta versión encontramos nuevas etiquetas y nuevos atributos para utilizar en nuestro documento web, pero además se ha definido una manera de como estructurar el contenido de nuestro documento.

En HTML5 podemos encontrar una serie de APIs que proporcionan una serie de recursos y facilidades para ayudar desarrollar aplicaciones web sin necesidad de ayuda externa más allá del lenguaje javascript en muchos casos. Por lo tanto podemos decir que HTML5 va de la mano de javascript en cuanto al desarrollo de aplicaciones se refiere.

Estas APIs deben ser implementadas por los propios navegadores para poder ser utilizadas. Se ha intentado que todos los navegadores las implementen basándose en la misma especificación, para que no ocurra como tantas otras veces que cada navegador haga la guerra por su cuenta, ya que los que lo pagan son los desarrolladores y posteriormente los usuarios ya que llegan a sitios donde su navegador no es compatible.

A pesar de estos esfuerzos, las partes de la implementación que quedaba en manos de los navegadores, en la mayoría de casos, cada uno lo ha hecho de una manera distinta, lo cual deriva en que la experiencia se ve afectada según el navegador que este utilizando, veremos que acontece el futuro.

Esta versión, que empezó a desarrollarse en 2004 y que todavía esta bajo esta condición, por lo que la especificación puede (y ha sucedido) estar sujeta a cambios, tanto en adición como sustracción de elementos, atributos y APIs. La estandarización oficial y la especificación de HTML5 vera la luz en el segundo cuatrimestre de 2014[3], según fuentes oficiales.

2.3. Semántica de HTML5

HTML5 nos propone una nueva manera de estructurar los documentos, y así dejar atrás la manera “tan rudimentaria” de diseccionar una web mediante etiquetas `div`, para luego insertar en ellas contenido. HML5 nos proporciona una serie de nuevas etiquetas con un significado específico para cada zona de nuestro documento web. Esta nueva forma de organizar la web proporciona al desarrollador elementos y atributos, así como la modificación de antiguos con nuevas funcionalidades. También hay algunos elementos y atributos que han dejado de tener implementación en esta nueva versión de HTML. Todo esto lo veremos con más detalle a continuación.

[2] Es un lenguaje de programación interpretado que esta implementado como parte de un navegador web. http://es.wikipedia.org/wiki/JavaScript.
[3] Anuncio oficial http://www.w3.org/News/2011.html#entry-9015.

2.3.1. Nueva estructuración del documento

Esta nueva versión de HTML nos proporciona una mejor manera para estructurar nuestro documento y distribuir la documentación a través de este. Anteriormente dividíamos las páginas de la siguiente manera:

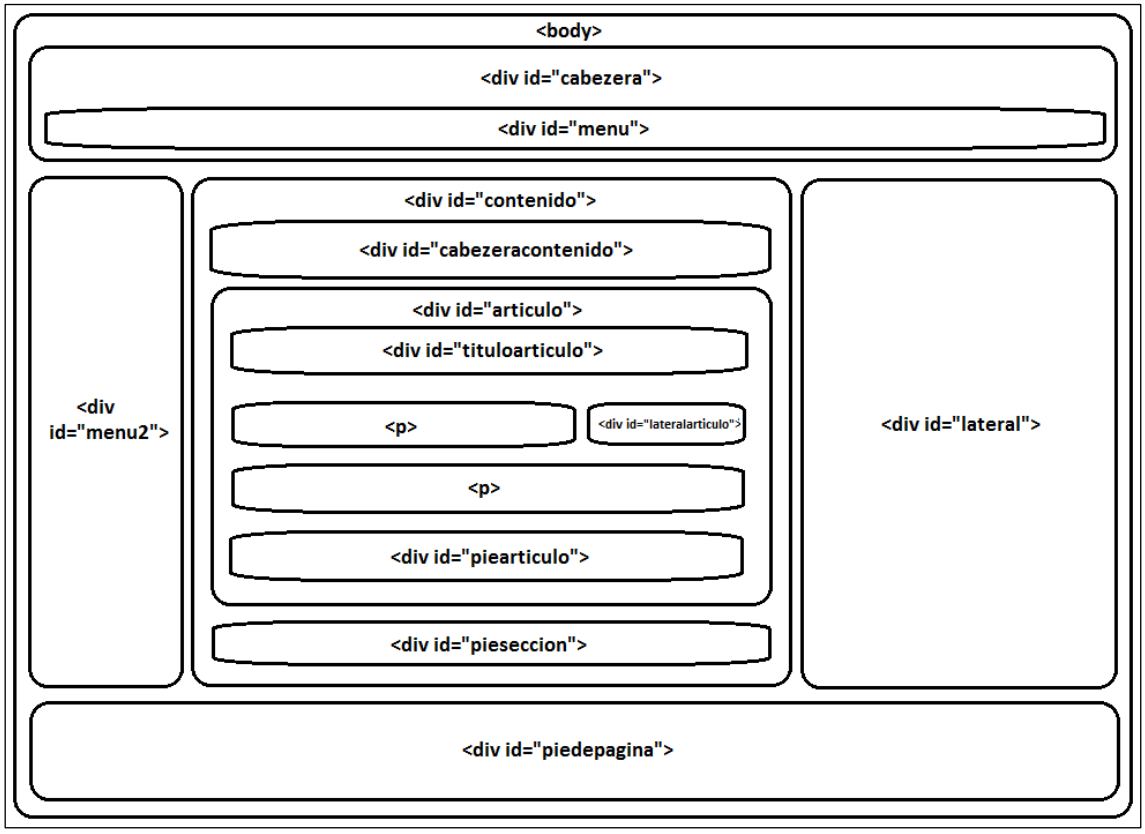

Figura2_01: Ejemplo de estructuración de una pagina usando HTML4.

En la versión anterior, generalmente, para distribuir el contenido se empleaba la etiqueta `div`. Para diferenciar cada una de estas etiquetas debíamos poner a cada una un identificador único, lo cual para páginas con gran contenido llegaba a ser un poco caótico. Luego este problema se reproducía en la hoja de estilos, ya que resultaba un tanto complejo seleccionar un determinado elemento. Esta etiqueta llego a este estándar como una solución temporal para poder distribuir el contenido a través de la web, una solución que ha tenido gran aceptación y utilización durante todo este tiempo y que seguirá teniendo en la nueva versión de HTML, aunque ya no para la estructura general del documento.

HMTL5 nos proporciona una nueva serie de etiquetas para poder realizar esta diferenciación de una manera más cómoda y sencilla. Un documento tipo en HTML5 tendría la siguiente estructura:

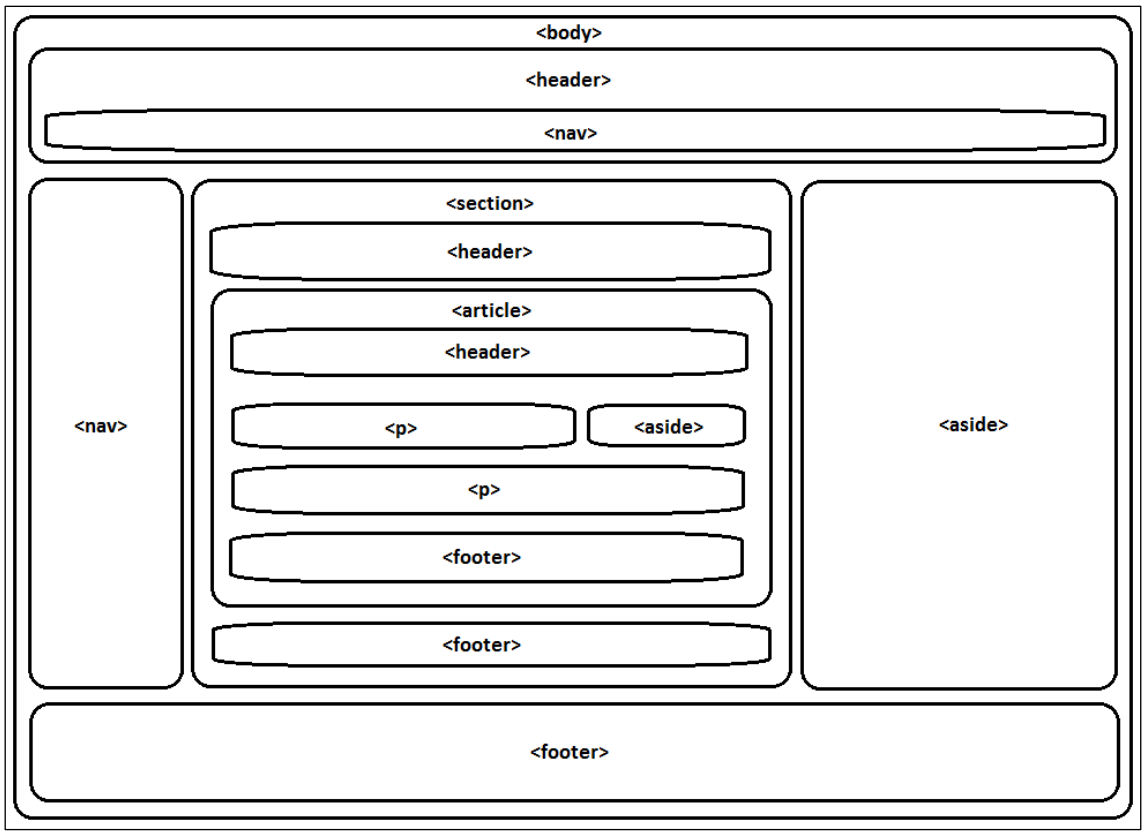

Figura2_02: Ejemplo de estructuración de una pagina utilizando HTML5.

Como podemos observar, hay una serie de nuevas etiquetas para cada parte de la hoja. La etiqueta `header` es para los elementos de la parte superior de la parte de donde este esta. La etiqueta `nav` es para los menús, se sitúa para definir la ubicación de un menú. La etiqueta `section` se utiliza para definir diferentes secciones de una pagina, que sean independientes de contenido entre si. `article` a su vez, define otro espacio que guarda relación con otros elementos `article` dentro de una mima sección. La etiqueta `aside` nos denota contenido lateral y finalmente `footer` se sitúa en la parte inferior de la sección en la que esta dispuesto.

A continuación veremos estas y otras nuevas etiquetas, donde las explicaremos con más detalle cada una de ellas.

2.3.2. Nuevas etiquetas

A continuación veremos y explicaremos una por una las nuevas etiquetas que nos proporciona HTML5.

`article` representa un bloque de información que es independiente del resto de contenido de la página. Por ejemplo, un elemento `article` seria una entrada en un blog, una noticia, un post en un foro,…

`aside` se usa para información que se posiciona a un lado. El contenido de esta información tiene que guardar relación con el elemento al cual es adjuntado.

`audio` se utiliza para definir un elemento de audio. Esta etiqueta se ve con más detalle en *2.5.1. Audio y video*.

`bdi` se utiliza para poder aislar una parte del texto que posteriormente puede ser formateado en una dirección distinta. Esta etiqueta es útil cuando se desconoce la direccionalidad del texto que emplean los usuarios de la web.

`canvas` se utiliza para indicar que contiene un grafico proporcionado por scripting. Esta etiqueta se ve con más detalle en *2.5.7. Canvas*.

`command` se utiliza para definir comandos dentro del documento, para llamar a una función cuando este es ejecutado.

`datalist` se utiliza para mostrar un conjunto de elementos que van dentro de una lista para el auto completado de un determinado campo de un formulario. Esta etiqueta se ve con más detalle en *2.4.1. Nuevos elementos de los formularios*.

`details` se utiliza para proporcionar más información acerca de un determinado elemento, pero solo se muestra cuando el usuario la solicita. O a menos que se declare abierta por defecto.

`embed` se utiliza para definir el emplazamiento de una aplicación externa o bien de contenido interactivo (algún tipo de plugin).

`figcaption` se usa para definir el titulo de una imagen perteneciente al elemento `figure`.

`figure` se usa para insertar imágenes relacionadas con el texto adyacente.

`footer` determina el pie de un articulo, sección o del propio documento. Determina el final de este aportando información en referencia del elemento al que pertenece.

`header` se utiliza del mismo modo que la anterior, solo que esta esta destinada a definir el encabezado de ese articulo, sección o documento.

`hgroup` se utiliza para agrupar elementos de titulo del documento `h1`-`h6`, y se utiliza cuando juntamos dos o más de etas etiquetas, es entonces cuando debemos agruparlas dentro de `hgroup`.

`keygen` se usa para generar un par de claves de cifrado cuando enviamos un formulario, una se almacena localmente y la otra se envía al servidor. Esta etiqueta se ve con más detalle en *2.4.1. Nuevos elementos de los formularios*.

`mark` se utiliza para marcar determinadas partes del texto, así luego poder resaltarlas con la ayuda de las hojas de estilo.

`menu` se utiliza para definir el menú que se mostrara cuando se solicite a través de un elemento con el atributo `contextmenu`.

`meter` se usa para proporcionar una descripción grafica de una medición escalar, donde conocemos el máximo. No debe utilizarse para representar un progreso de una tarea o

proceso, pues para ello ya hay una etiqueta especifica. Esta etiqueta se ve con más detalle en *2.4.1. Nuevos elementos de los formularios*.

`nav` se utiliza para determinar una sección de vínculos de navegación, también conocidos como enlaces. No todos los enlaces deben estar bajo esa etiqueta, solamente el bloque principal de enlaces, es decir, el menú o menús.

`output` se utiliza para mostrar resultados referentes a entradas en los formularos, tales como operaciones aritméticas o la muestra de valores. Esta etiqueta se ve con más detalle en *2.4.1. Nuevos elementos de los formularios*.

`progress` se utiliza para representar gráficamente el porcentaje de un determinado proceso o tarea. Esta etiqueta se ve con más detalle en *2.4.1. Nuevos elementos de los formularios*.

`rp` se encuentra dentro del elemento `ruby` y sirve para mostrar texto en aquellos navegadores que no soportan anotaciones en ruby.

`rt` se encuentra dentro del elemento `ruby` y se utiliza para proporcionar información acerca de la pronunciación de el contenido del elemento ruby.

`ruby` se utiliza en la tipografía del este asiático, para mostrar la pronunciación de los caracteres.

`section` determina una parte del contenido de un documento web. Gracias a esta etiqueta diferenciamos cada una de las partes de la web, pudiendo estructurarla de una manera mucho más clara y sencilla.

`source` se utiliza para especificar medios multimedia múltiples, donde el navegador es el que elige en función del códec que cada uno soporta. Se pueden definir medios tanto audio como video. Esta etiqueta se ve con más detalle en *2.5.1. Audio y video*.

`summary` se utiliza dentro del elemento `details`, y es el encabezamiento visible sobre el cual clicamos para ver la información detallada.

`time` se utiliza para codificar las horas o fechas de nuestro documento en un lenguaje más cómodo para nuestro navegador o nuestro código, con el cual podemos trabajar mucho mejor al no tener que tratarlo como una simple cadena de texto.

`track` se utiliza para definir los subtítulos u otros elementos de texto para nuestros archivos de multimedia, tanto audio como video. Esta etiqueta se ve con más detalle en *2.5.1. Audio y video*.

`video` se utiliza para definir un elemento de video. Esta etiqueta se ve con más detalle en *2.5.1. Audio y video*.

`wbr` se utiliza para hacer un salto de línea dentro de nuestro texto. Si es cierto que el navegador lo hace automáticamente, pero si queremos controlarlo para que se produzca en un determinado sitio, debemos utilizar esta etiqueta.

2.3.3. Etiquetes modificadas

En esta parte vamos a detallar aquellas etiquetas que ya existían en la versión anterior y que han sufrido algún tipo de modificación, bien sea en su especificación como en su utilización.

`a` esta etiqueta ha perdido algunos atributos obsoletos, esto son: `charset`, `coords`, `name`, `rev` y `shape`. Por el contrario se han añadido dos de nuevos, por un lado el atributo `media` que sirve para especificar para que medios o dispositivos ha sido optimizado el siguiente link, y por otro `type` el cual sirve para especificar que tipo MIME del archivo que nos vamos a encontrar en ese enlace.

`b` se utiliza de la misma manera que en su versión anterior, pero en HTML5 no se debe usar para cambiar el formato de texto solamente, si no que para remarcar las palabras sobre las que queremos hacer un fuerte énfasis, creando un efecto de llamar la atención visualmente, pero no así en la voz o el estado anímico.

`cite` tiene la misma utilización que en la anterior versión, pero en HMTL5 solo sirve para citar títulos de obras, ya no sirve para citar cualquier otro tipo de material.

`hr` esta etiqueta se utiliza de manera similar, pero, en vez de ser una marca en el texto, a partir de ahora definirá un párrafo o párrafos como de temática diferente y los cuales podrán ser diferenciados en estilo mediante las hojas de estilo en cascada.

`i` al igual que ocurre con b, este elemento ya no sirve solo para convertir el texto a itálico, si no que se debe utilizar cuando escribimos algo fuera de la prosa del texto, es decir, frases en otros idiomas, términos técnicos,…

`input` esta etiqueta se ha visto mejorada notoriamente en cuanto a sus atributos se refieren, se han añadido los siguientes: `autocomplete`, `autofocus`, `form`, `formaction`, `formentype`, `formmethod`, `formnovalidate`, `formtarget`, `height`, `list`, `max`, `min`, `multiple`, `pattern`, `placeholder`, `required`, `step` y `width`. Todos ellos proporcionan de nuevas funcionalidades a esta etiqueta. Esta etiqueta se ve con más detalle en *2.4.3. Nuevos tipos de input*.

`small` esta etiqueta ahora ya no sirve para reducir el tamaño del texto respecto del original, si no que cambia su utilización. Ahora debe utilizarse como lo que conocemos comúnmente como letra pequeña, alertas, avisos y demás jerga legal; pero solo la utilizaremos en fracciones pequeñas de texto, si tenemos varios párrafos o incluso paginas enteras de documentación ya no utilizamos esta etiqueta.

`u` anteriormente se usaba para subrayar, pero mucha gente se confundía ya que esperaba un enlace; ahora su utilización queda reservada para marcar faltas de ortografía en un texto, ya que es la manera mas común de hacerlo de los editores de texto. También se usa para subrayar nombres propios en chino ya que es la manera correcta de hacerlo.

2.3.4. Etiquetes eliminadas

Las siguientes etiquetas han dejado de tener soporte en esta nueva versión de HTML5, por haber quedado obsoletas, o bien por no tener contextualización en el marco tecnológico actual.

`acronym`

`applet`

`basefont`

`big`

`center`

`dir`

`font`

`frame`

`frameset`

`noframes`

`strike`

`tt`

2.3.5. Cambios en la estructura del documento.

A continuación veremos la manera correcta de definir nuestro documento, para hacerlo correctamente, debemos hacerlo de la siguiente manera:

Doctype

En anteriores versiones de HMTL veíamos largas cadenas que nos definían el tipo de documento:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

Figura2_03: Ejemplo de codificación del documento en HTML4.

En la nueva versión se ha pasado a:

```
<! DOCTYPE HTML>
```

Figura2_04: Ejemplo de codificación del documento en HTML5.

Este es uno de los cambios más llamativos de HTML5, en esta nueva versión hemos dejado atrás esa larga cadena que contenía tanta información, así como el numero de versión. Este aspecto se ha decidió, así no porque no se vaya a evolucionar mas HTML5, si no porque los

navegadores modernos mostraran correctamente los elementos de HTML hasta donde tengan implementado, y donde no lleguen, sencillamente, no lo mostraran. Por lo tanto es irrelevante conocer la versión del documento.

Si el `DOCTYPE` no esta declarado correctamente, los nuevos selectores de CSS3 no funcionaran.

Charset

Para definir la codificación de caracteres anteriormente utilizábamos:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

Figura2_05: Ejemplo de la codificación de caracteres en un documento HTML4.

Donde ahora para hacer lo mismo utilizamos simplemente:

```
<meta charset="utf-8">
```

Figura2_06: Ejemplo de la codificación de caracteres en un documento HMLT5.

Links a la hoja de estilos y a archivos javascript

Anteriormente para hacer referencia a cada uno de estos elementos utilizábamos:

```
<link href="main.css" rel="stylesheet" type="text/css" >
<script src=" modernizr.js" type="text/javascript"></script>
```

Figura2_07: Ejemplo de enlace una la hoja de estilos y a un archivo javascript en HTML4.

En esta versión se ha prescindido del elemento type, lo cual nos deja algo tal que así:

```
<link href="main.css" rel="stylesheet" >
<script src=" modernizr.js"></script>
```

Figura2_08: Ejemplo de enlace a una hoja de estilos y a un archivo javascript en HTML5.

Un documento completo con todos elementos seria de la siguiente manera:

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Titulo del documento</title>
        <link href="main.css" rel="stylesheet" >
        <script src="modernizr.js"></script>
    </head>
    <body>
        <p>Contenido</p>
    </body>
</html>
```

Figura2_09: Ejemplo de documento completo en HTML5.

2.3.6. Nuevos atributos globales

Estos nuevos atributos, al igual que ocurre con los de la versión anterior, se pueden definir en cualquier elemento del documento. De la anterior versión heredamos `accesskey`, `class`, `dir`, `id`, `lang`, `style`, `tabindex` y `title`; a los cuales añadimos los siguientes atributos:

`contenteditable` Este atributo se utiliza para hacer que el contenido del elemento en el cual se alberga pueda ser modificado por el usuario. Esta etiqueta se ve con más detalle en *2.5.2. Edicion en el documento*.

`contextmenu` Este atributo se utiliza para referenciar a un menú proporcionado por el autor, este menú únicamente se muestra cuando es solicitado por el usuario a través del propio elemento que contiene este atributo. El menú a mostrar debe declararse en la etiqueta menú.

`data-*` Este atributo nos permite especificar datos en el elemento que lo contiene. Posteriormente podemos a acceder a los que nos interesen desde javascript, permitiéndonos hacerlo así de una manera mucho más sencilla de acceder. El nombre debe ser la cadena data- y al menos un carácter, seguidamente especificamos el valor que contendrá.

`draggable` y `dropzone` Estos atributos se usan para declarar zonas de contenido arrastrable hacia fuera o hacia dentro. Esta etiqueta se ve con más detalle en *2.5.4. Arrastrar y soltar*.

`hidden` Este atributo se usa para esconder el elemento de la vista del usuario, también lo hace a la vista de lectores de pantalla, pero no para secuencias de comando o controles de formulario. Este atributo es booleano, por lo tanto se puede hacer aparecer o desaparecer con ayuda de algún script.

`role y aria-*` Este atributo se utiliza para dar un extra de accesibilidad a nuestra web, proporcionando a los usuarios con discapacidades una ayuda extra para entender mejor los distintos elementos y el contenido del documento. Estos atributos nos permiten dar una información extra al usuario del contenido que alberga el elemento o para informar del propósito de este. Esta información será interpretada por los lectores de pantalla. Aria al igual que data debe tener uno o más caracteres la denominación que le demos, y posteriormente indicamos el valor que tiene.

`spellchek` Este atributo se usa para corregir la ortografía de un determinado texto gracias a nuestro navegador, los errores aparecen en un subrayado rojo, muy similar a los editores de texto de escritorio.

`translate` Esta atributo se utiliza para marcar que parte de nuestro documento queremos que se traduzca o no mediante herramientas para ello. Las herramientas para traducir páginas web protegerán de traducción aquellos elementos que tengan este atributo a falso.

2.4. Formularios HTML5

En la versión HTML5 encontramos resueltos algunos problemas o dificultades, que nos surgían a la hora de implementar formularios en nuestro sitio web, en anteriores versiones de HTML.

Antes de HTML5 los formularios eran bastante primitivos y rudimentarios, es por eso que los formularios un poco más complejos, requerían un buen nivel de javascript (u otros lenguajes) trabajando detrás. Con estos lenguajes ayudábamos al usuario a la hora de introducir datos en los formularios, y a los desarrolladores para verificar que los datos introducidos por estos eran correctos. HTML5 ha hecho un esfuerzo importante en mejorar este aspecto, proporcionando una batería de elementos y atributos nuevos. Esta mejora, que no solo ayuda a los programadores web a verificar formularios utilizando únicamente código HTML, si no que proporciona una ayuda a los usuarios cuando se encuentran delante de estos. Algunos navegadores ya tienen implementados prácticamente todas estas novedades, especialmente webkit y todavía en mayor dedicación Opera. El problema hoy por hoy es que, primeramente, no todos los navegadores tienen implementado el 100% de las nuevas funcionalidades, pero las que si tienen todos implementadas, no todos lo hacen de la misma manera, proporcionando resultados diferentes según el navegador, la cual cosa dificulta que una web proporcione la misma experiencia al usuario en todos los navegadores, veremos que acontece el futuro.

2.4.1. Nuevos elementos de los formularios

En la versión de HTML5 se han añadido los siguientes elementos para los formularios: `output`, `datalist`, `meter`, `progress` y `keygen`. Veámoslos más detalladamente da uno de ellos:

Output

El elemento `output` (salida) representa un valor determinado de un formulario, su uso más común, es la numeración de un rango o el resultado de una operación aritmética de varios valores del formulario.

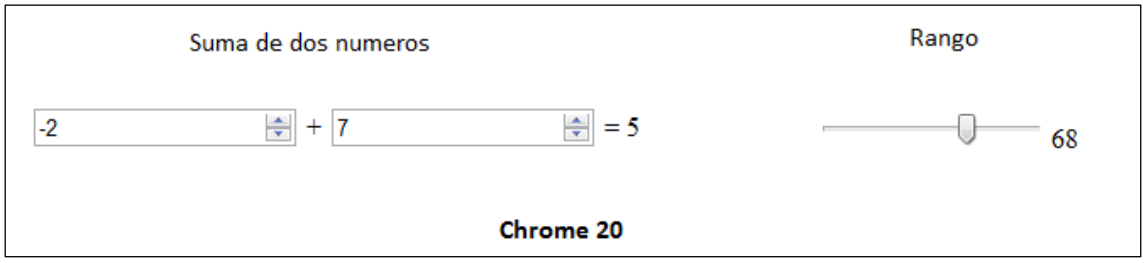

Figura2_10: Ejemplo de utilización del elemento output utilizando el navegador Chrome.

En la Figura2_10 hemos implementado dos Outputs, a la izquierda tenemos la suma de dos números y a la derecha un rango. En cuanto hacemos un cambio en cualquiera de los inputs automáticamente el resultado del output cambia y vemos los resultados instantáneamente en la web.

El código para implementar el primer output seria el siguiente:

```
<form>
    <input name=num1 id=num1 type=number> +
    <input name=num2 id=num2 type=number> =
    <output for="num1 num2"></output>
        <script>
        var form = document.querySelector('form'),
            num1 = form.num1,
            num2 = form.num2,
            output = document.querySelector('output');
        [].forEach.call(document.querySelectorAll('input'), function(el) {
            el.addEventListener('blur', function() {
                output.value = ~~num1.value + ~~num2.value;
            }, false);
        });
    </script>
</form>
```

Figura2_11: Ejemplo de código de la utilización del elemento output. Ejemplo sacado de [1].

Y para implementar el segundo:

```
<form id="the-form">
    <input type="range" id="range" name="range">
    <output for="range"></output>
    <script>
        (function () {
            var theForm = document.getElementById("the-form");
            if ("oninput" in theForm) {
                theForm.addEventListener("input", function () {
                    output.value = range.value;
                }, false);
            }
        })();
    </script>
</form>
```

Figura2_12: Ejemplo de código de la utilización del elemento output. Ejemplo sacado del libro WAY J. ,2012, Decoding HTML5, Rockable.

Datalist

El elemento `datalist` (lista de datos) proporciona una ayuda extra para rellenar un campo de texto, ofreciendo al usuario varias opciones predefinidas para completarlo automáticamente; Estas opciones predefinidas son sugerencias por lo que el usuario puede escribir el mismo lo que desee si su respuesta no esta dentro de las opciones dadas.

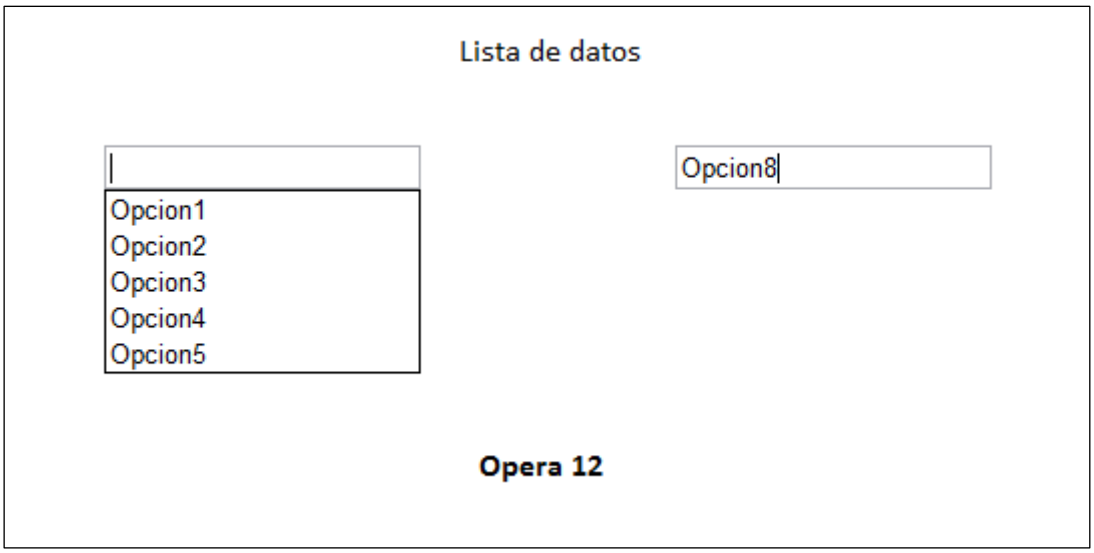

Figura2_13: Ejemplo de utilización del elemento datalist utilizando el navegador Opera.

Como vemos en la Figura2_13, al clicar sobre el campo del formulario, nos aparece una lista de posibles opciones, pero si no es ninguna de estas, podemos escribirla nosotros mismos.

Para implementar esto hemos utilizado el siguiente código:

```
<form>
    <input type="text" name="Opciones" list="data-list">
        <datalist id="data-list">
            <option value="Opcion1"></option>
            <option value="Opcion2"></option>
            <option value="Opcion3"></option>
            <option value="Opcion4"></option>
            <option value="Opcion5"></option>
        </datalist>
</form>
```

Figura2_14: Ejemplo de código de la utilización del elemento datalist.

Progress

El elemento `progress` (progreso) representa gráficamente el porcentaje de una tarea, como por ejemplo una carga, una descarga o el proceso de rellenado de un formulario, …

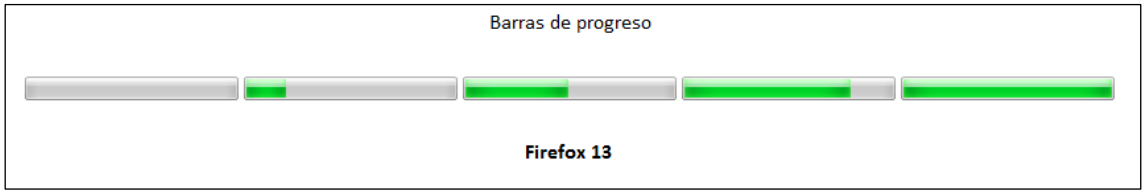

Figura2_15: Ejemplo de utilización del elemento progress utilizando el navegador Firefox.

En la Figura2_15 podemos apreciar distintos tipos de porcentaje: 0%, 20%, 50%, 80% y 100% respectivamente.

Para implementar esto hemos utilizado el siguiente código:

```
<form>
    <progress max=10 value=0></progress>
    <progress max=10 value=2></progress>
    <progress max=10 value=5></progress>
    <progress max=10 value=8></progress>
    <progress max=10 value=10></progress>
</form>
```

Figura2_16: Ejemplo del código de la utilización del elemento progress.

Meter

El elemento `meter` (medidor) proporciona información grafica acerca de un elemento que queramos medir, proporcionándonos una ayuda extra si queremos tener controlados tanto valores máximos como mínimos.

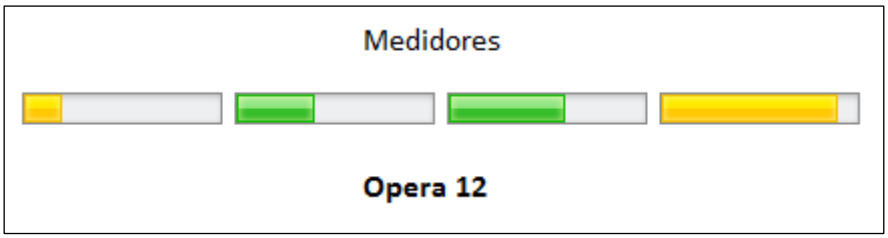

Figura2_17: Ejemplo de utilización del elemento meter utilizando el navegador Opera.

En la figura2_17 podemos apreciar que aparecen resaltados tanto los valores altos como los mínimos, eso sucede porque hemos definido los valores por debajo del 30% y 80% como mínimos y máximos respectivamente, y una vez superados nos proporcionan este efecto.

Hemos utilizado el siguiente código para implementar el medidor:

```
<form>
    <meter low="3" optimum="5" high="8" min="0" max="10" value="2"></meter>
    <meter low="3" optimum="5" high="8" min="0" max="10" value="4"></meter>
    <meter low="3" optimum="5" high="8" min="0" max="10" value="6"></meter>
    <meter low="3" optimum="5" high="8" min="0" max="10" value="9"></meter>
</form>
```

Figura2_18: Ejemplo de código de la utilización del elemento meter.

Keygen

El elemento `keygen` (generador de claves) proporciona una manera segura de autenticar a los clientes. Este elemento genera dos claves, una privada que es almacenada localmente, y otra publica que enviamos en el formulario. Una vez enviada la clave pública podremos generar certificados del cliente pudiendo autenticar así a los usuarios posteriormente de una manera segura.

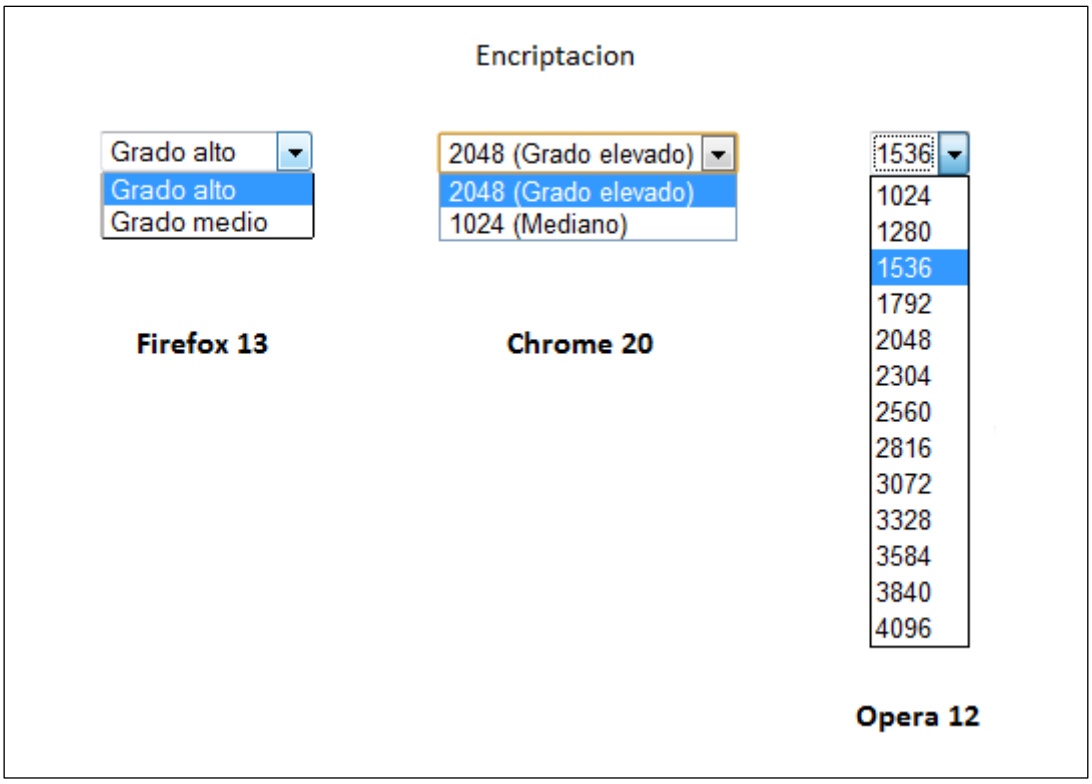

Figura2_19: Ejemplo de utilización del elemento keygen en diferentes navegadores.

Como podemos ver en la imagen, cada navegador lo implementa de diferente manera, en especial opera el cual nos permite elegir entre un buen repertorio el tamaño de nuestras claves, llegándonos a ofrecer incluso 4096 bytes.

Para implementar esto hemos utilizado el siguiente código:

```
<form>
  <keygen name="security" />
</form>
```

Figura2_20: Ejemplo de código de la utilización del elemento keygen.

2.4.2. Nuevos atributos en los formularios HTML5

Los nuevos atributos en esta nueva versión de HTML5 mejoran el funcionamiento, no solo de los nuevos elementos, vistos anteriormente, si no que a los antiguos también. Proporcionando así nuevas características y funcionalidades a todos los elementos de un formulario. Los distinguiremos en dos tipos, los atributos que van dentro de la etiqueta `form`, y los que van dentro de la etiqueta `input` que son la mayoría.

Dentro de la etiqueta form:

Autocomplete

El atributo autocomplete (completar automáticamente) proporciona la posibilidad de permitir al navegador rellenar los campos automáticamente. Esta opción se activa y desactiva mediante las palabras “on” y “off”, como podemos apreciar a continuación:

```
<form autocomplete=”on”></form>
```

Figura2_21: Ejemplo de código de la utilización del atributo autocomplete.

Novalidate

El atributo novalidate (no validar) deshabilita la función de validar un formulario antes de ser enviado. Se utiliza cuando queremos admitir todo tipo de respuestas en nuestro formulario, cuando el usuario escribe datos en un campo no adaptado para ello. Como por ejemplo aceptar letras en un campo únicamente destinado a la introducción de números.

```
<form novalidate=”novalidate”></form>
```

Figura2_22: Ejemplo de código de la utilización del atributo novalidate.

Dentro de la etiqueta input:

Autocomplete

El atributo autocomplete (completar automáticamente) aquí funciona exactamente de la misma manera que en la etiqueta `form`, pero esta vez solo tiene repercusión en el input al cual es aplicado. No tiene mucho sentido aplicar el autocompletado campo por campo, pero si que lo tiene en caso de ser un formulario que se autocomplete, bloquear el auto completado de determinados campos, como por ejemplo contraseñas o números de cuenta o de tarjetas de crédito. A continuación vemos como funciona:

```
<input type="password" autocomplete="off">
```

Figura2_23: Ejemplo de código de la utilización del atributo autocomplete.

Autofocus

El atributo autofocus (enfoque automático) proporciona un enfoque automático al cargar la pagina, cuando se trata de introducción de datos por teclado. El cursor ya esta listo para escribir en el. Solo podemos poner este atributo en un único elemento `input` de un único `form` de cada pagina.

```
<input type=”text” autofocus>
```

Figura2_24: Ejemplo de código de la utilización del atributo autofocus.

Step

El atributo step (intervalo) nos permite definir un intervalo de valores para un determinado campo. Si con step definimos el numero 5, solo serán validos 5 y todos sus múltiplos.

```
<input type="number" step="5">
```

Figura2_25: Ejemplo de código de la utilización del atributo autofocus.

Dirname

El atributo dirname (direccion de texto) se utiliza cuando la introducción de texto se realiza de derecha a izquierda, si el atributo dirname esta presente cuando se envía el formulario se invierte el orden de la cadena para poder utilizarla con la misma implementación como si una de tipo occidental se tratara.

```
<input type=”text” dirname>
```

Figura2_26: Ejemplo de código de la utilización del atributo dirname.

Form

El atributo form (formulario) dentro de un input se utiliza, cuando ese input no se encuentra cerrado entre las etiquetas del propio formulario. Mediante este atributo podemos indicar a que formulario pertenece apuntando como atributo el identificador del formulario en el cual queremos integrar. Del mismo modo, el mismo campo puede ser referido a mas de un formulario dejando un espacio en blanco entre cada uno de los identificadores.

```
<input type="text" name="segundo_apellido" form="form2">
```

Figura2_27: Ejemplo de código de la utilización del atributo form.

Width y height

Los atributos widht y height (ancho y altura respectivamente) nos permiten definir el tamaño que tendrá un determinado elemento del formulario. Estos atributos dentro del formulario solo se utilizan en inputs de tipo `image`.

```
<input type="image" src="submit.gif" alt="submit" width="48" height="48">
```

Figura2_28: Ejemplo de código de la utilización de los atributos widht y height.

Min y Max

Los atributos min y max (mínimo y máximo respectivamente) nos permiten definir el valor máximo y mínimo de un determinado campo. Con estos atributos solo se pueden definir campos contables, es decir, de tipo `number`, `range`, `date`, `datetime`, `datetime-local`, `month`, `time` y `week`.

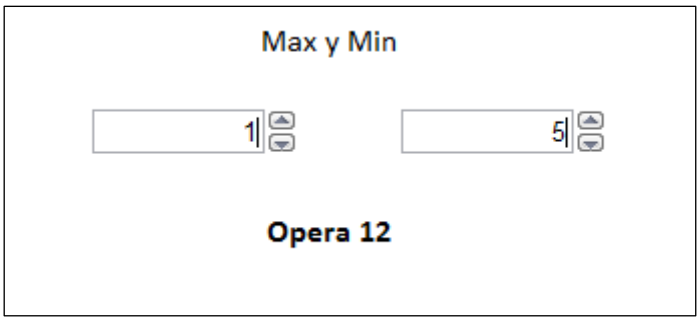

Figura2_29: Ejemplo de utilización de los atributos min y max utilizando el navegador Opera.

Como vemos en la figura2_29, en ambos campos tenemos definido como maximo y minimo 5 y 1 respectivamente; nuestro navegador nos permite subir y bajar los valores mediante las flechas, pero no nos deja sobrepasar el límite. Esto lo hemos conseguido mediante el siguiente código.

```
<input type="number" min="1" max="5" />
```

Figura2_30: Ejemplo de código de la utilización de los atributos max y min.

Multiple

El atributo multiple (múltiple) permite introducir mas de un resultado en un mismo campo de un formulario, separado por comas. Este atributo solo es posible utilizarlo en inputs de tipo `file` y de tipo `email`, para poder insertar varios archivos o diferentes cuentas de correo respectivamente, en el mismo formulario.

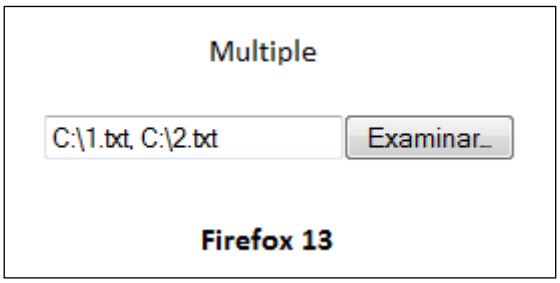

Figura2_31: Ejemplo de utilización del atributo multiple utilizando el navegador Firefox.

En esta figura2_31 vemos como podemos seleccionar dos archivos a la vez separados con una coma. Con el botón examinar brinda la posibilidad al usuario de hacerlo gráficamente y entonces es el propio navegador el que introduce las comas de separación.

```
<input type="file" multiple="multiple">
```

Figura2_32: Ejemplo de código de la utilización del atributo multiple.

Pattern

El atributo pattern (patrón) nos permite regular el valor introducido en el input, de este modo el usuario sabe si esta esta bien introducido al momento y no es necesario esperar a validar el

formulario. Si además usamos el atributo `title` poniendo las instrucciones ayudaremos al usuario a la hora de rellenar correctamente el campo.

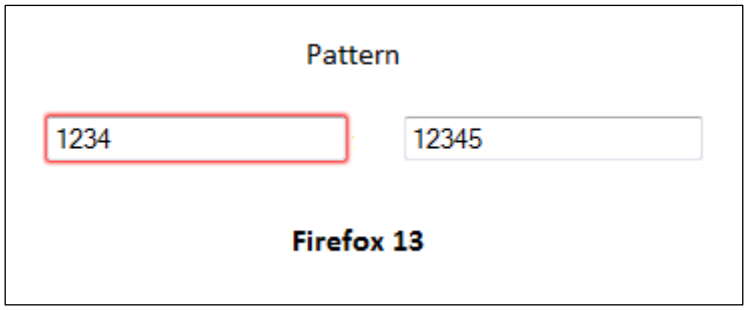

Figura2_33: Ejemplo de utilización del atributo pattern utilizando el navegador Firefox.

Como vemos en la figura2_33, nuestro navegador nos resalta que no hemos introducido correctamente los datos, una vez introducido correctamente según el pattern desaparece la marca roja.

```
<input type="number"title="Codigo zip:inserta 5 numeros" pattern="[0-9]{5}">
```

Figura2_34: Ejemplo de código de la utilización del atributo pattern.

Placeholder

El atributo placeholder (marcador de posición) se utiliza para dar una ayuda al usuario a la hora de introducir los datos en un determinado campo. Muestra el texto del placeholder dentro del input en un color claro que cuando empezamos a escribir desaparece. Este atributo solo puede definirse en los inputs de tipo `text`, `search`, `url`, `tel`, `email` y `password`.

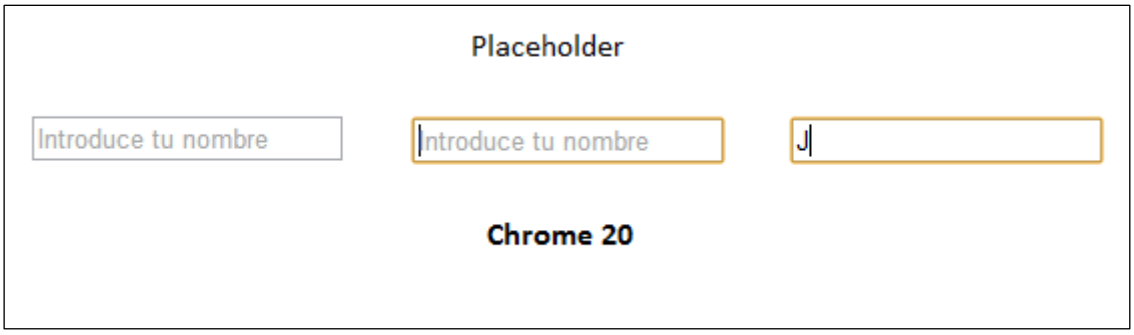

Figura2_35: Ejemplo de utilización del atributo placeholder utilizando el navegador Chrome.

Como vemos en la imagen, nos aparece en el cuadro el texto de ayuda que desaparece en cuanto empezamos a escribir.

```
<input type="text" placeholder="Introduce tu nombre">
```

Figura2_36: Ejemplo de código de la utilización del atributo placeholder.

Required

El atributo required (obligatorio) obliga al usuario a rellenar ese campo antes de poder mandar el formulario. Este atributo solo puede insertarse en inputs de tipo `text`, `search`, `url`, `tel`, `email`, `password`, `date` y derivados, `number`, `checkbox`, `radio` y `file`.

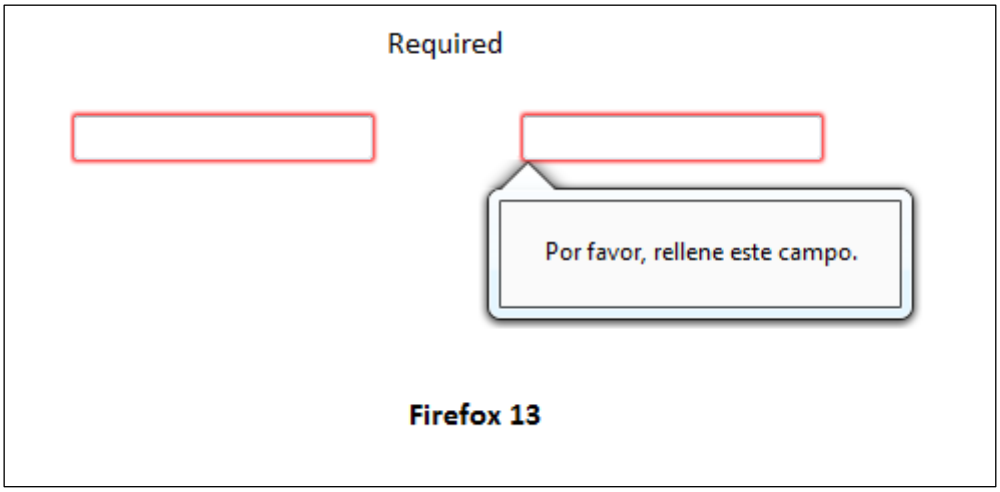

Figura2_37: Ejemplo de utilización del atributo required utilizando el navegador Firefox.

Como apreciamos en la figura2_37, en este caso, Firefox es el único navegador que resalta el cuadro solo por estar vacío, pero obviamos esta advertencia e intentamos enviar el formulario nos salta el mensaje de la parte derecha, al igual que los otros navegadores que también tienen implementada esta característica.

```
<input type="text" required>
```

Figura2_38: Ejemplo de código de la utilización del atributo required.

Atributos que redefinen atributos del form

Son un conjunto de atributos que todos ellos tienen las mismas características, se utilizan en un input de tipo submit y redefinen atributos del form justo antes de ser enviado.

`formaction:` se utiliza cuando queremos cambiar el atributo action.

`fromenctype:` se utiliza cuando queremos cambiar el atributo enctype.

`formmethod:` se utiliza cuando queremos cambiar el atributo method.

`formnovalidate:` se utiliza cuando queremos cambiar el atributo novalidate.

`formnotarget:` se utiliza cuando queremos cambiar el atributo target.

A continuación vemos como se aplican cada uno de ellos, en el mismo orden.

```
<form action="form.php" method="get">
    <input type="submit" value="Enviar">

    <input type="submit" formaction="admin.php" value="Como administrador">

    <input type="submit" formenctype="multipart/form-data" value="Multipart/form-data" />

    <input type="submit" formmethod="post" formaction="formpost.asp" value=" con POST" />

    <input type="submit" formnovalidate="formnovalidate" value="Sin validacion" />

    <input type="submit" formtarget="_blank" value="Enviar nueva ventana" />
</form>
```

Figura2_39: Ejemplo de código de la utilización de los diferentes atributos de modificación de otros atributos del form.

Nótese que el atributo formenctype solo puede ser utilizado con el método post.

2.4.3. Nuevos tipos de input

En esta nueva versión de HTML5 además de las antiguas etiquetas `button`, `checkbox`, `file`, `hidden`, `image`, `password`, `radio`, `reset`, `submit` y `text` se han añadido 13 nuevas: `datetime`, `datetime-local`, `date`, `month`, `week`, `time`, `number`, `range`, `tel`, `email`, `url`, `color` y `search`. Estos nuevos tipos de input permiten una introducción de datos de manera controlada por el propio navegador, por lo tanto no requiere programación extra para tratarlos posteriormente. Los navegadores que todavía no soportan estos nuevos tipos de entrada tratan los campos como si de texto normal se trataran. Cabe destacar la dedicación de Opera que es el navegador que cuenta con más soporte para este nuevo tipo de datos.

Datetime

Este tipo de input nos permite introducir una fecha y una hora, que además nos guarda la zona horaria en la que estamos.

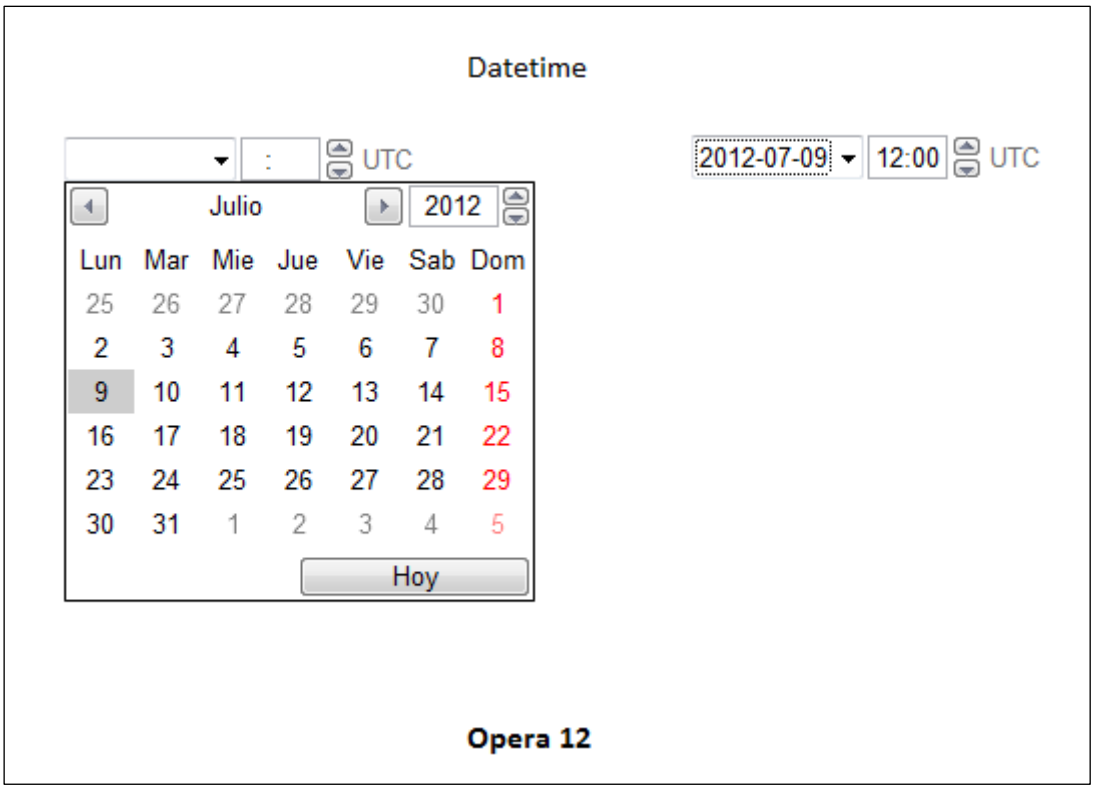

Figura2_40: Ejemplo de utilización del input de tipo datetime utilizando el navegador Opera.

Como podemos apreciar en la figura2_40, al pulsar sobre el campo, en opera se nos despliega un calendario donde podemos elegir el día, y una vez elegido la hora se pone por defecto a 12:00 y mediante las flechas seleccionamos la hora que queremos.

```
<input type="datetime">
```

Figura2_41: Ejemplo de código de la utilización del input de tipo datetime.

Datetime-local

Funciona exactamente igual, pero este tipo no guarda la zona horaria.

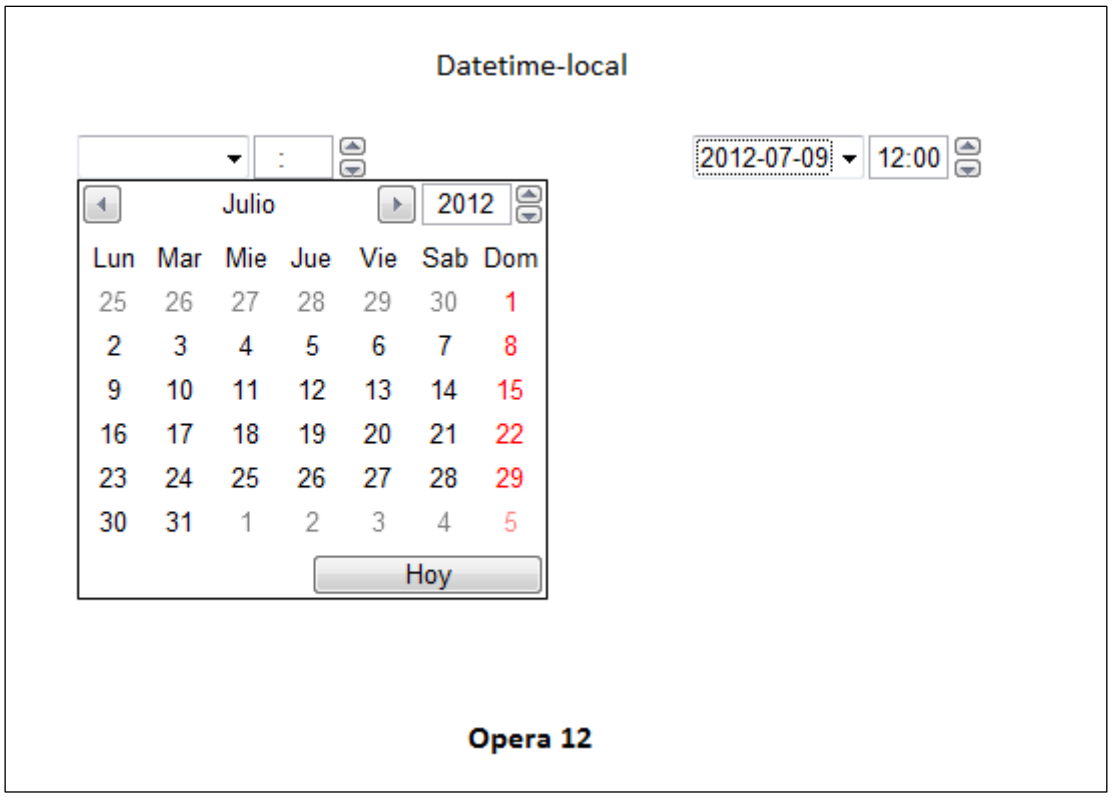

Figura2_42: Ejemplo de utilización del input de tipo datetime-local utilizando el navegador Opera.

Como se aprecia en la figura2_42 desaparece la marca del UTC, pero el funcionamiento es exactamente el mismo.

```
<input type="datetime-local">
```

Figura2_43: Ejemplo de código de la utilización del input de tipo datetime-local.

Date

Este tipo de entrada nos permite seleccionar un día del calendario desplegable, el funcionamiento es el mismo que en los casos anteriores, pero únicamente contando con un día concreto.

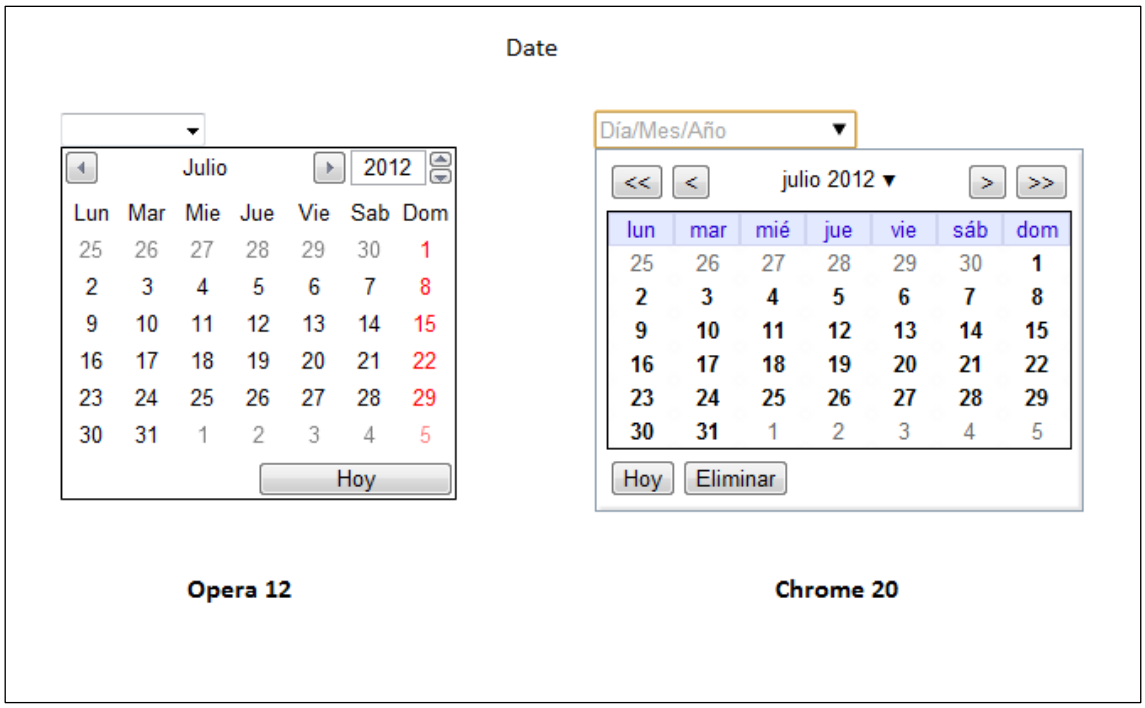

Figura2_44: Ejemplo de utilización del input de tipo date utilizando los navegadores Opera y Chrome.

En la figura2_44 se aprecia que solo disponemos del campo para la fecha.

```
<input type="date">
```

Figura2_45: Ejemplo de código de la utilización del input de tipo date.

Month

Este tipo de entrada de datos nos permite elegir un mes determinado.

Figura2_46: Ejemplo de utilización del input de tipo month utilizando el navegador Opera.

Como se aprecia en la figura2_46, el calendario de opera es el mismo que en los demás casos, pero al pasar el cursor por encima nos marca como selección todos los días del mes, y para elegir un mes debemos utilizar las flechas para cambiar, lo cual denota que no esta muy optimizado; pero únicamente opera tiene implementada esta característica.

```
<input type="month">
```

Figura2_47: Ejemplo de código de la utilización del input de tipo month.

Week

Este tipo nos permite seleccionar una semana del calendario desplegable.

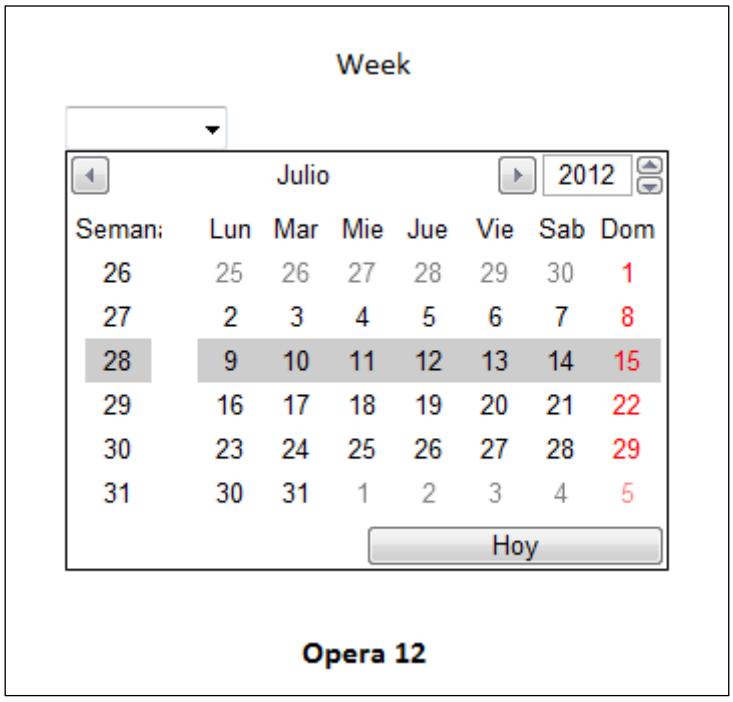

Figura2_48: Ejemplo de utilización del input de tipo week utilizando el navegador Opera.

Como vemos en la figura2_48, con el cursor nos selecciona una semana entera, también destacar que en la parte izquierda aparecen las semanas numeradas.

```
<input type="week">
```

Figura2_49: Ejemplo de código de la utilización del input de tipo week.

Time

Este tipo de input nos permite introducir una hora determinada.

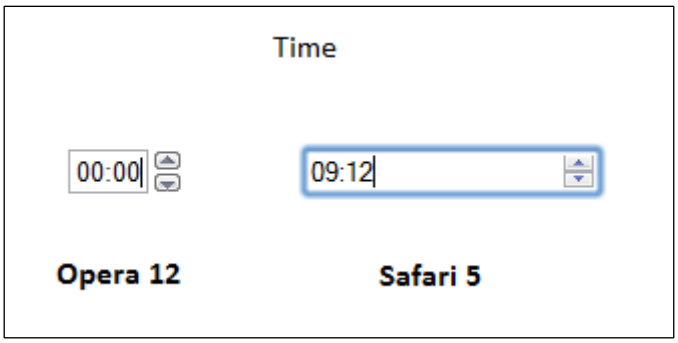

Figura2_50: Ejemplo de utilización del input de tipo time utilizando los navegadores Opera y Safari.

En la figura2_50 se aprecia como aparecen en ambos campos en la parte derecha flechas para poder cambiar la hora, también destacar que opera nos da por defecto las 00:00 y safari, en cambio, la hora actual.

```
<input type="time">
```

Figura2_51: Ejemplo de código de la utilización del input de tipo time.

Number

El tipo de entrada number nos permite definir campos que solo puedan contener números, proporcionando a los usuarios una ayuda en forma de flechas para aumentar o disminuir la cantidad; con el atributo `step`, al usar las flechas el valor saltara en la cantidad definida.

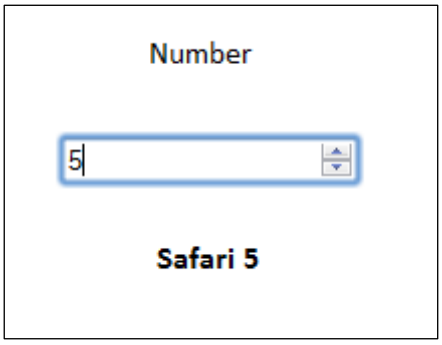

Figura2_52: Ejemplo de utilización del input de tipo number utilizando el navegador Safari.

En la figura2_52 podemos ver que al pulsar las flechas incrementa el numero, si definimos el atributo step lo hará en ese valor, en nuestro caso lo hemos definido a 5 y no negativos con el atributo min a 0.

```
<input type="number" step="5" min="0">
```

Figura2_53: Ejemplo de código de la utilización del input de tipo number.

Range

Este tipo nos representa gráficamente un elemento movible que simboliza el rango de valores definidos en el input.

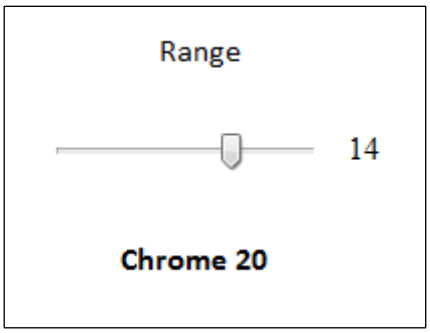

Figura2_54: Ejemplo de utilización del input de tipo range utilizando el navegador Chrome.

Como vemos en la figura2_54, podemos desplazar la barra de derecha a izquierda por todos los valores permitidos en el rango. Para visualizar el resultado nos hemos ayudado del elemento output.

```
<input type="range" min="0" max="20" step="2">
```

Figura2_55: Ejemplo de código de la utilización del input de tipo range.

Tel

Este nuevo tipo nos permite registrar números de teléfono, destacar que ningún navegador tiene implementada esta característica actualmente.

```
<input type="tel">
```

Figura2_56: Ejemplo de código de la utilización del input de tipo tel.

Color

Este tipo nos permite seleccionar un color sin necesidad de conocer su numeración, ya que el navegador nos proporcionara ayuda grafica.

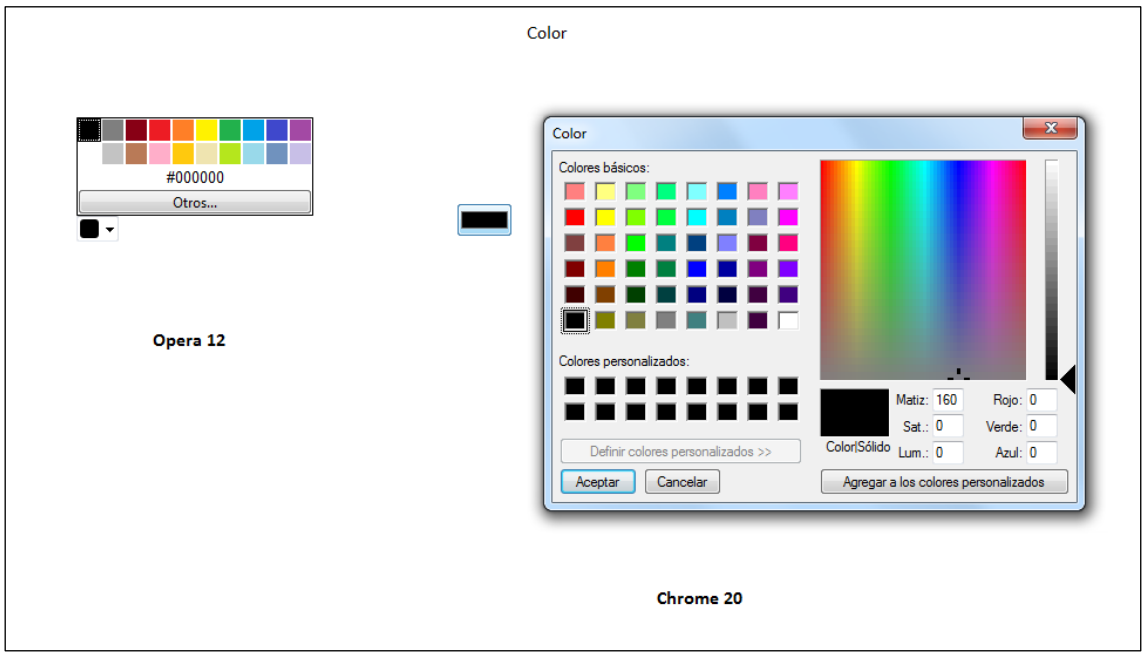

Figura2_57: Ejemplo de utilización del input de tipo color utilizando los navegadores Opera y Chrome.

Como vemos en la figura2_57, los navegadores nos proporcionan ayuda (junto con el sistema). Destacar que Opera nos da una batería de colores principales, si pulsamos en el cuadro Otros… se nos despliega el mismo cuadro del sistema que en Chrome desde un principio.

```
<input type="color">
```

Figura2_58: Ejemplo de código de la utilización del input de tipo color.

Email

Este tipo nos permite definir campos que solo acepten direcciones de correo validas sin necesidad de escribir código extra.

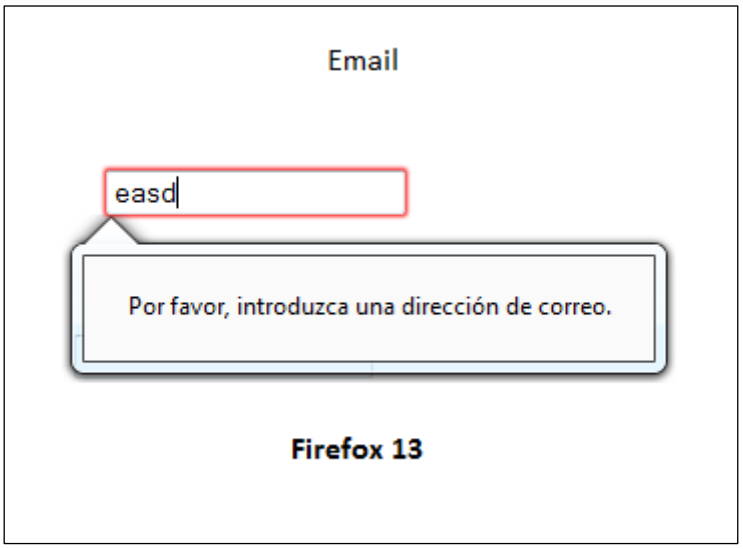

Figura2_59: Ejemplo de utilización del input de tipo email utilizando el navegador Firefox.

Aquí en la figura2_59 vemos como nuestro navegador nos resalta que la dirección de correo introducida no es valida. Resaltar que únicamente comprueba que la cadena introducida sea algo@algo, así que si queremos restringir algún dominio concreto debemos aplicar el atributo pattern.

```
<input type="email">
```

Figura2_60: Ejemplo de código de la utilización del input de tipo email.

URL

Este elemento nos permite restringir campos que contengan una dirección url valida.

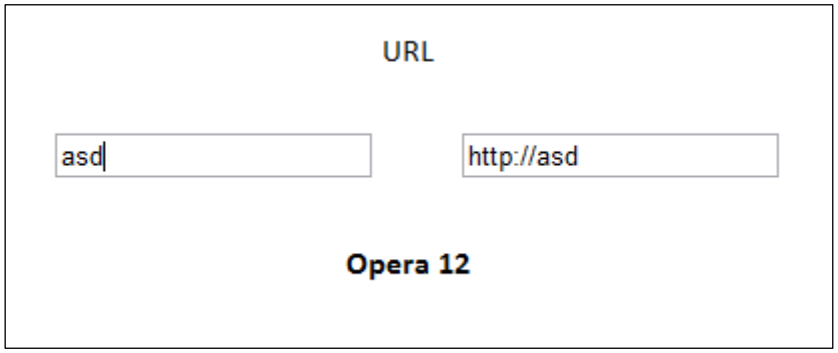

Figura2_61: Ejemplo de utilización del input de tipo url utilizando el navegador Opera.

Como se aprecia en la figura2_61 el navegador Opera si introducimos una url no valida, cuando acabamos de editar ese campo nos agrega automáticamente la cadena “http://” al principio. Igual que el caso anterior este atributo no tiene en cuenta los dominios, solo comprueba el principio de la cadena introducida.

```
<input type="url">
```

Figura2_62: Ejemplo de código de la utilización del input de tipo url.

Search

Este elemento nos permite definir cuadros de búsqueda, y que el propio navegador nos ayude con sus propias herramientas.

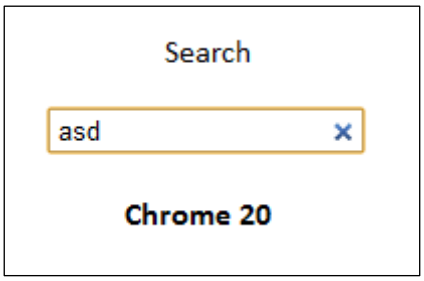

Figura2_63: Ejemplo de utilización del input de tipo search utilizando el navegador Chrome.

Como vemos en la imagen webkit, en este caso chrome más concretamente, tiene implementada esa aspa que aparece cuando escribimos alguna cosa en ese campo para vaciarlo. Además si añadimos los atributos results y autosave en el input, nos aparecerá un icono de búsqueda a la izquierda del campo y podremos acceder a antiguas búsquedas.

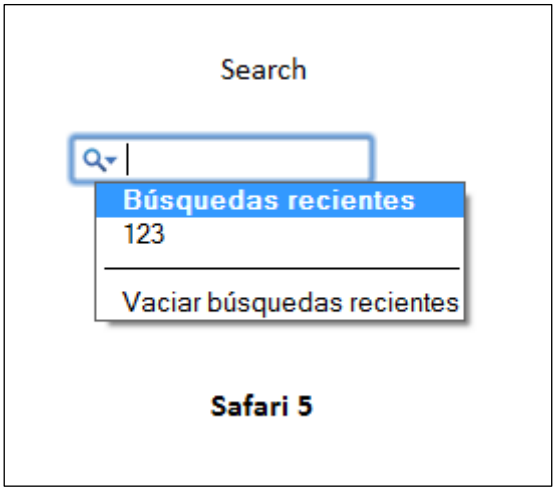

Figura2_64: Ejemplo de utilización del input de tipo search utilizando el navegador Safari.

Como podemos ver en la figura2_64, con estos dos atributos nuestro navegador nos ha implementado un cuadro de búsqueda con historial incluido.

```
<input type="search" results="5" autosave="search-queries">
```

Figura2_65: Ejemplo de código de la utilización del input de tipo search.

2.5. HTML5 APIS

A continuación veremos las APIs que incorpora HTML5 para facilitar el uso de ciertas características en la web. Todas estas herramientas proporcionan al desarrollador un amplio marco donde poder desarrollar aplicaciones web con HTML5 de la mano de Javascript. A continuación veremos las APIs que hay implementadas en HTML5.

2.5.1. Audio y video

La nueva versión de HTML5 nos permite incrustar videos y audios en nuestra web de forma nativa, sin necesidad de utilizar ningún otro código más allá del propio HTML. Todos los navegadores actuales soportan la inserción de video y audio, el problema esta en los codecs que estos utilizan. A continuación vemos una tabla, indicado para cada navegador que códec soporta:

VIDEO				AUDIO				
	ogg	webm	mp4	wav	ogg	mp3	aac	
Chrome	√	√	X	√	√	√	√	Chrome
Opera	√	√	X	√	√	X	X	Opera
Safari	X	X	√	√	X	√	√	Safari
Internet Explorer	X	X	√	X	X	√	√	Internet Explorer
Firefox	√	√	X	√	√	X	X	Firefox

Figura2_66: Tabla con los codecs que soporta cada navegador.

Cada navegador tiene implementado su propio reproductor, con sus controles predeterminados, siendo unos más completos que otros. Los controles los podemos editar, modificar o añadir usando un lenguaje externo, como javascript por ejemplo. A continuación veremos como insertar un video en nuestro documento:

```
<video controls width="650px" height="365px">
    <source src="Earth-HD.mp4" type="video/mp4" />
    <source src="Earth-HD.webm" type="video/webm" />
</video>
```

Figura2_67: Ejemplo de código de la inserción de un video en el documento web.

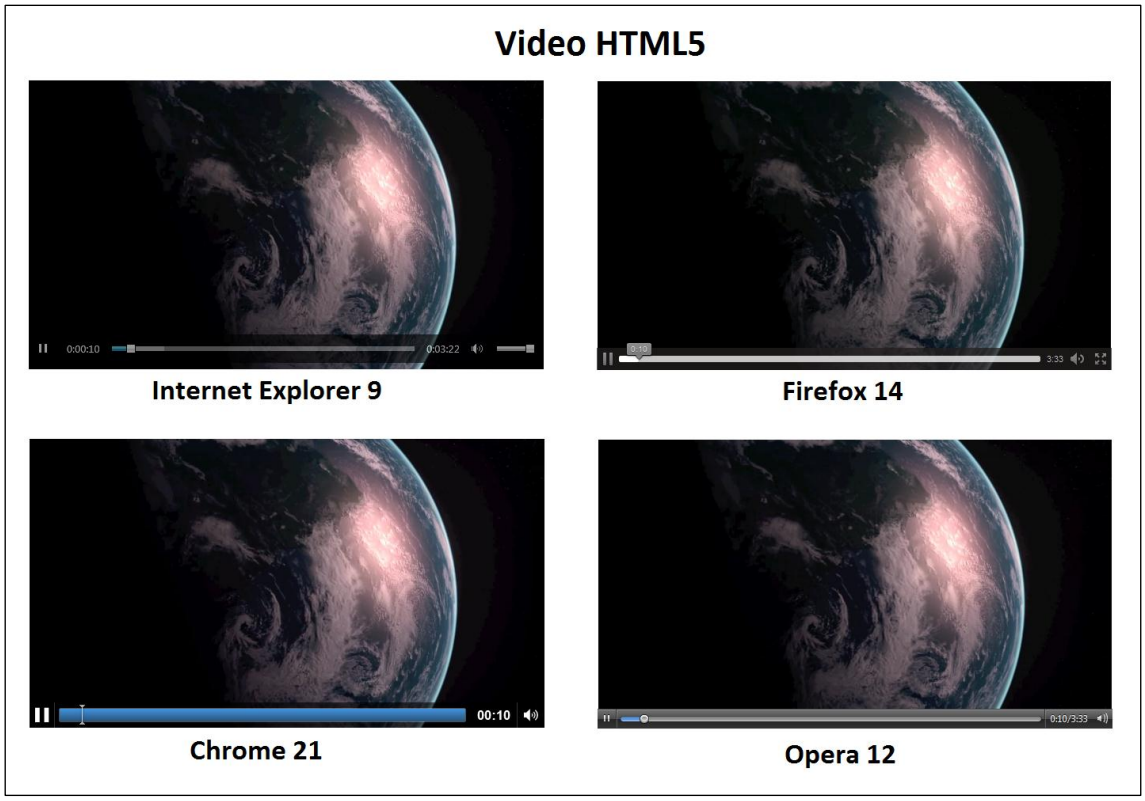

Figura2_68: Ejemplo de inserción de video utilizando distintos navegadores.

Como podemos apreciar en la figura2_68, cada navegador tiene implementado su propio reproductor y estos no se aparecen ni en aspecto ni en funcionalidades unos a otros.

Lo mismo ocurre cuando implementamos un elemento de audio:

```
<audio controls>
        <source src="Earth_HD.mp3" type="audio/mpeg" />
        <source src="Earth_HD.wav" type="audio/wav" />
</audio>
```

Figura2_69: Ejemplo de código de la inserción de audio en el documento.

El resultado es el siguiente dependiendo del tipo de navegador, al igual que en el caso anterior.

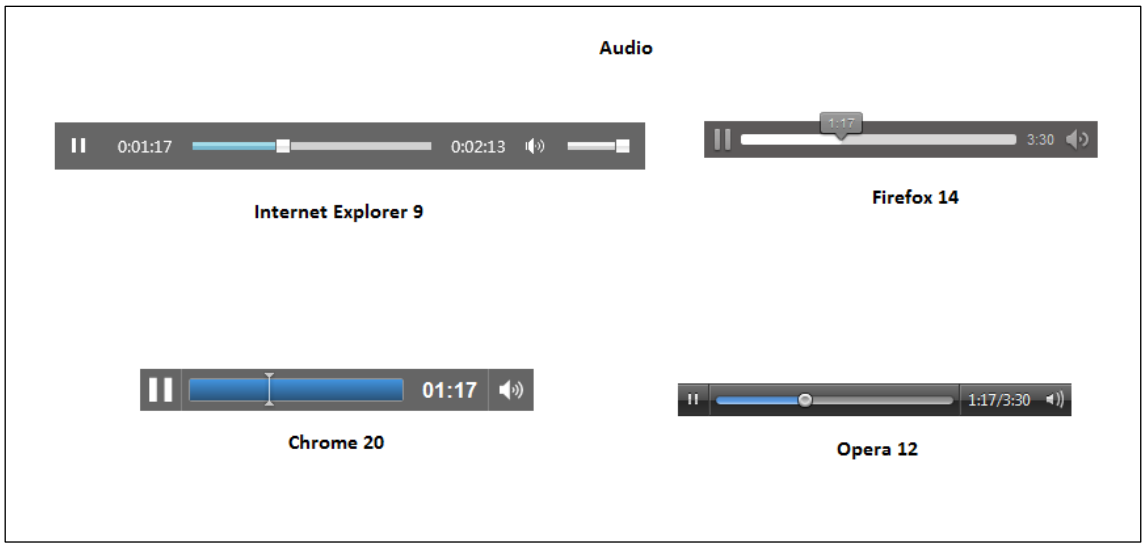

Figura2_70: Ejemplo de la inserción de audio utilizando distintos navegadores.

Todas estas nuevas funcionalidades y características, las podemos explotar un poco mas utilizando estos atributos y propiedades que veremos a continuación.

Primero veremos los atributos relacionados con la etiqueta video:

`autoplay` se utiliza para empezar a reproducir en cuanto este cargado en la página.

`controls` se utiliza para especificar si se muestran los controles sobre el reproductor, el botón de play y pausa, el control de volumen, la barra de tiempo,… cada navegador incluye los suyos propios.

`height` se utiliza para especificar la altura que tendrá el reproductor de video en la web.

`loop` se utiliza para indicar que el video vuelva a empezar cuando llegue al final.

`muted` se utiliza para especificar que la salida de audio del video será silenciada.

`poster` se utiliza para definir una imagen, la cual se mostrara en el emplazamiento que ocupe el video en la web hasta antes de este se empiece a reproducir. Debemos indicar la URL de la imagen.

`preload` se utiliza para especificar si el video se empezara a cargar con la carga de la pagina o no lo hará hasta que se pulse el botón play. También podemos definir que únicamente cargue los metadatos al inicio.

`src` se utiliza para especificar la url del video que queremos reproducir , si no queremos utilizar la etiqueta source.

`width` se utiliza para especificar el ancho que tendrá el reproductor de video en la web.

Seguidamente veremos los atributos que disponemos en la etiqueta audio:

`autoplay` se utiliza para empezar a reproducir el elemento tan pronto como se haya cargado en la página.

`controls` se utiliza para habilitar los controles de play, pause, control de volumen, barra de tiempo,… al usuario.

`loop` se utiliza para determinar que una pista de audio se vuelva a empezar reproducir cuando esta acabe.

`preload` se utiliza para especificar si se cargaran datos cuando se cargue la página. Podemos especificar que cuando se cargue la página se cargue la pista de audio, los matadatos o nada.

`src` se utiliza para especificar la URL de nuestro archivo de audio si no utilizamos al etiqueta source.

A continuación veremos los atributos del elemento source, que son comunes tanto en el elemento video como en audio.

`media` se utiliza para definir el comportamiento que tendrá el archivo de video o el audio al reproducirse. Este atributo nos permite cambiar la forma de ver u oir el contenido dependiendo del tipo de dispositivo, y sus características, que el usuario utilice para reproducir el contenido.

`src` se utiliza para especificar la URL del elemento de video o audio que vayamos a insertar.

`type` se utiliza para indicar el tipo MIME del archivo que contiene este elemento.

En ambos elementos, disponemos del elemento track que nos permite definir subtítulos o cualquier otro archivo de texto que deba ser mostrado durante la reproducción del propio elemento. A continuación veremos los atributos que de este elemento track:

`default` se utiliza para marcar un elemento como predeterminado para utilizar.

`kind` se utiliza para definir el tipo de archivo que disponemos, este puede ser `captions`, `chapters`, `descriptions`, `metadata` o `subtitles`.

`label` se utiliza para definir el titulo de ese elemento de texto para cuando se muestre la lista de cuales están disponibles en el navegador.

`src` se utiliza para especificar la URL del archivo de texto que queremos insertar.

`srclang` se utiliza para indicar el idioma del archivo de texto añadido utilizando el “lenguage code” (el dos letras: en, es, ca, it,…).

2.5.2. Edición en el documento (Inline editing API)

Esta nueva API nos brinda la posibilidad de editar un texto haciéndolo directamente sobre el en la propia la web. Para poder utilizarla basta con declarar el atributo `contenteditable` en el elemento al cual queremos dotar de dicha característica.

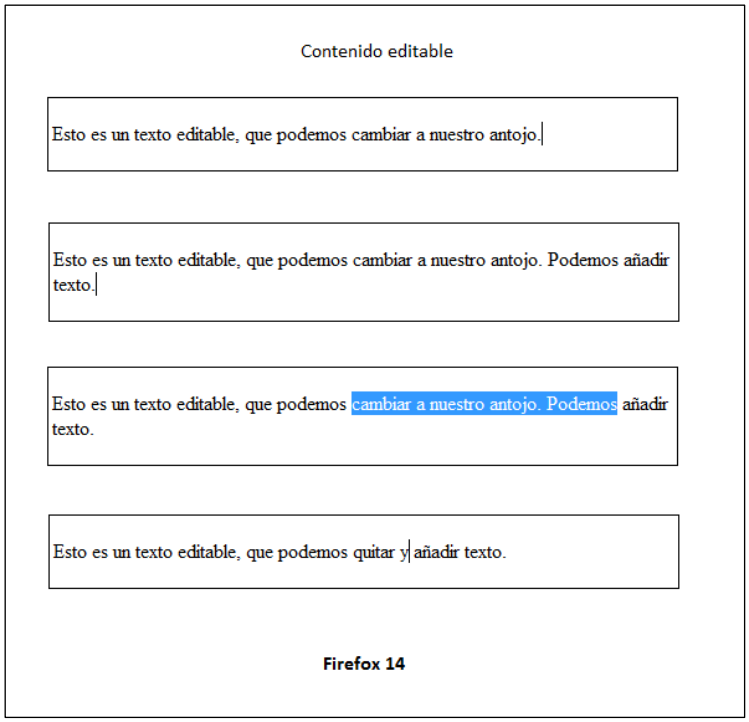

Figura2_71: Ejemplo del uso de la API de contenido editable utilizando el navegador Firefox.

Para realizar esto hemos utilizado el siguiente código:

```
<div contentEditable="true">
   <p>Esto es un texto editable, que podemos cambiar a nuestro antojo.</p>
</div>
```

Figura2_72: Ejemplo de código de la utilización de la API de contenido editable.

Con esta API únicamente podemos modificar el texto directamente de la web, pero con la ayuda de javascript podemos acceder a este texto modificado. Además podemos ir un poco más allá y si además usamos hojas de estilo y de unos simples botones podemos implementar rápida y sencillamente un editor de texto en nuestra web.

2.5.3. Historial del navegador (Browser History API)

Esta API de HTML5 se nos permite utilizar el historial de sesión de nuestro navegador de una nueva manera totalmente distinta. Mediante esta API podemos guardar en cambios en el estado del documento y no únicamente las paginas visitadas. En cada evento que ocurra en nuestro documento podemos almacenarlo, y luego recorrerlos todos simplemente recorriendo el historial de nuestro navegador. El historial de sesión debemos utilizarlo como una pila, al cual podemos añadir y quitar siempre de la parte superior de esta, así como desplazarnos por los elementos, pudiendo modificarlos.

Para utilizar esta API debemos emplear los siguientes métodos que ejecutaremos desde javascript:

`history.back()` se utiliza para ir a la pagina anterior del historial.

`history.forward()` se utiliza para ir a la pagina siguiente del historial.

`history.go(entero)` se utiliza para saltar el numero, entero que indiquemos, de paginas en el historial. Si usamos valores negativos, el salto se realiza hacia atrás. Utilizar history.go(-1) y history.go(1) es equivalente a usar history.back() y history.forward() respectivamente.

`history.pushState (datos, titulo, url)` se utiliza para añadir una nueva página en el historial. El primer parámetro nos permite almacenar cualquier objeto de javascript. El segundo parámetro es el titulo que llevara esa nueva página. Y el tercer elemento es la url de la página, este valor puede omitirse ya que en caso de hacerlo por defecto almacenara la url actual.

`history.replaceState( datos, titulo, url)` se utiliza de la misma manera que `history.pushState()` pero este en vez de añadir una pagina nueva, remplaza a la actual pagina.

`history.length` se utiliza para obtener el número de elementos de nuestro historial de sesión incluyendo la página actual.

`history.state` se utiliza para obtener la página de historial actual (datos, titulo y url). Este método lo podemos utilizar cuando no utilicemos evento popstate.

`window.onpopstate` este evento se dispara cada vez que nos desplazamos a través de la pila del historial de sesión (mediante `history.back()` o `history.forward()` o equivalentes), proporcionándonos el estado (datos, titulo y url) de la pagina sobre la que nos hayamos desplazado.

2.5.4. Arrastrar y soltar (Drag and drop API)

En HTML5 se ha desarrollado esta API que consiste en desplazar elementos a través de un documento web con el ratón. La API funciona clicando sobre el elemento determinado, arrastrarlo y soltarlo en otro lugar, produciendo unos resultados deseados por el programador de la web. Esto anteriormente requería un esfuerzo importante por parte de los desarrolladores del sitio web, gracias a esta API esta tarea se ha simplificado en gran medida. A continuación explicaremos como utilizarla.

En primer lugar, para dotar a un elemento de la cualidad de ser desplazado, debemos agregar a la declaración de dicho elemento el atributo `dragable=”true”`. Esto nos permitirá captar los movimientos que haga el usuario sobre este elemento, mediante los siguientes eventos:

`dragstart` se lanza cuando la acción de arrastrar empieza, desde el primer movimiento.

`dragenter` se lanza cuando el elemento que estamos arrastrando entra en la zona donde puede ser soltado.

`dragover` se lanza cuando el elemento que estamos arrastrando se sitúa encima del elemento sobre el que podría soltarse.

`dragleave` se lanza cuando el elemento que estamos arrastrando se sale de la zona donde puede ser soltado (primero ha tenido que entrar en la zona).

`drag` se lanza cuando (y durante) estamos arrastrando un elemento.

`drop` se lanza cuando un elemento es soltado dentro de la zona donde puede ser soltado.

`dragend` se lanza cuando soltamos un elemento que previamente hemos estado arrastrando.

Los eventos referentes al arrastrado del elemento se deben gestionar desde el propio elemento, los eventos referentes al soltado deben ser gestionados sobre los elementos sobre los cuales queremos depositar tal elemento.

Una vez tenemos capturada la acción que queremos, cuando ejecutamos una función podemos transferir datos al objeto evento de las siguientes maneras:

`dataTransfer.setData (tipo, datos)` se usa para almacenar los datos que queramos transmitir. Estos datos deben estar asociados con el elemento que estamos arrastrando. Los tipos debemos indicarlo en una cadena para indicar el tipo de datos en formato MIME.

`dataTransfer.clearData (tipo)` se usa para borrar el elemento guardado de ese tipo.

`dataTransfer.setDragImage (imagen, x, y)` se usa para definir una imagen propia que queremos que se muestra mientras arrastramos un elemento. Los valores x e y son las compensaciones referentes a los eje de x y de y en referencia puntero y al centro de la imagen.

`dataTransfer.effectAllowed` se usa para determinar que operaciones podremos realizar sobre los datos. Si usamos copiar, significa que los datos se duplicaran; si elegimos mover, significa que copiaremos los datos y los originales se borraran; si elegimos link, significa que se establecerá algún tipo de conexión entre los datos de ambos elementos. Podemos definir `none`, `copy`, `move`, `link`, `copyMove`, `copyLink`, `linkMove` y `all`.

Cuando la acción de arrastrar se ha llevado a cabo, depositando el elemento dentro del emplazamiento esperado, ahora podemos utilizar los siguientes métodos en ese evento:

`dataTransfer.effectDrop` se usa para determinar las operaciones que podremos realizar sobre los datos. En este caso solo podremos definir none, copy, move y link.

`dataTransfer.types` se usa para obtener una lista de los tipos que tiene almacenados el evento.

`dataTransfer.Files` se usa para obtener una lista de archivos que han sido almacenados en el event.

`dataTransfer.getData (tipo)` se usa para obtener los datos del tipo de datos que le indiquemos como parámetro.

2.5.5. Archivos (File API)

En HTML5 se ha implementado esta API que nos permite la interacción de ficheros entre la web y nuestro sistema. Primero de todo empezaremos por ver como podemos leer (o cargar) ficheros del sistema a nuestro documento web. Para ello podemos utilizar, bien el método de drag and drop visto en la explicacion de la API anterior o bien una entrada de un formulario de tipo `file`. Para poder acceder a los ficheros debemos hacerlo creando eventos y utilizando las funciones callback.

En caso de utilizar un formulario para cargar los archivos, mediante:

```
event.target.file[i]
```

Figura2_73: Ejemplo de código para la obtención de elementos cargados mediante formulario.

Podremos acceder a los ficheros que hayamos seleccionado, si por el contrario hemos utilizado la API de drag and drop deberemos acceder a los archivos mediante:

```
event.dataTransfer.files[i]
```

Figura2_74: Ejemplo de código para la obtención de elementos cargados mediante la API de drag and drop.

Donde i, en ambos casos, representa cada elemento cargado en caso de haber habilitado la función de cargar múltiples archivos. Podemos ver los atributos de cada archivo utilizando los siguientes métodos:

`name` se utiliza para obtener el nombre del archivo, incluyendo su extensión.

`size` se utiliza para obtener el tamaño del archivo en Bytes.

`type` se utiliza para obtener el tipo de MIME que es el archivo.

Si queremos utilizar estos archivos dentro de nuestro documento, debemos crear un nuevo objeto de la clase fileReader:

```
var reader = new fileReader();
```

Figura2_75: Ejemplo de código para la obtener un nuevo objeto de la clase FileReader.

Una vez hecho esto podemos utilizar los métodos que este nuevo objeto nos pone a nuestra disposición para poder utilizar los archivos cargados.

`readAsText(archivo)` se utiliza para leer el texto que contiene un archivo cargado. Si el archivo cargado no es de tipo texto, no se podrá leer el archivo.

`readAsDataURL(archivo)` se utiliza para leer el archivo como una URL, obteniendo la dirección URL de este.

`readAsBinaryString(archivo)` se utiliza para leer el archivo como una cadena binaria.

`readAsArrayBuffer(archivo)` se utiliza para leer el archive como un arrayBuffer.

Para poder controlar mejor nuestro objeto lector, podemos utilizar los siguientes eventos:

`onload` se lanza cuando FileReader ha acabado de leer correctamente un archivo.

`onloadstart` se lanza cuando la acción de leer acaba de empezar.

`onloadend` se lanza cuando FileReader ha acabado de leer con o sin errores.

`onprogress` se lanza periódicamente para obtener el progreso de lectura de un archivo.

`onerror` se lanza cuando se produce cualquier error en la lectura de un archivo.

`onabort` se lanza cuando el proceso de lectura de un fichero es interrumpida.

Ahora, si en vez de leer, queremos descargar un archivo a nuestro sistema mediante drag and drop, debemos implementarlo de la misma manera a como lo haríamos si lo arrastráramos dentro de nuestro propio documento. Con la diferencia que el elemento con el atributo `dragable` declarado, debe contener también el atributo `data-downloadurl` con el valor

```
“Tipo MIME del archivo:Nombre del archivo:URL absoluta al archivo”
```

Figura2_76: Ejemplo de cadena de texto para introducir en el atributo data-downloadurl.

Cuando utilizamos el evento para cargar el archivo en el evento debemos indicar que es de tipo `downloadurl`, y como datos indicar el dataset definido como downloadurl, de la siguiente manera:

```
event.dataTransfer.setData('DownloadURL', this.dataset.downloadurl);
```

Figura2_77: Ejemplo de código de la carga de datos en el evento.

Tras lanzarse el evento, se iniciará la descarga del archivo hacia nuestro sistema. Esta característica solamente esta disponible en google chrome.

2.5.6. Posicionamiento geográfico (Geolocation API)

Esta nueva versión de HTML5 nos proporciona la capacidad de localizarnos geográficamente. Esta propiedad esta disponible en todos los navegadores actuales de cualquier plataforma, aunque se obtiene más precisión en dispositivos con GPS incorporado.

Para utilizar este api, debemos llamar al método:

```
navigator.geolocation.getCurrentPosition(parametro1, parametro2, {opciones})
```

Figura2_78: Ejemplo de código de la obtención de la posición geográfica.

Ambos parámetros son funciones callback, es decir, las funciones que se ejecutaran después de llamar a dicho método. El parámetro1 llamara a la función con ese nombre pasando un objeto con la información acerca de la posición del usuario, el parámetro2 es opcional y se ejecutara únicamente cuando haya un error, y esta tendrá como parámetros el propio error para poder tratarlo. Las opciones que podemos definir son opcionales, por defecto todas son 0.

También tenemos el método:

```
navigator.geolocation.watchPosition(parametro1, parametro2, {opciones})
```

Figura2_79: Ejemplo de código de la obtención de la posición geográfica.

Este método funciona de la misma manera que `getCurrentPosition`, pero este método se utiliza cuando queramos capturar cambios de posiciones ya que la función de callback se llamara periódicamente con los datos actualizados. Para parar las llamadas a la función de callback, debemos utilizar el método `clearwatch(parametro).` Pasándole como parámetro el objeto sobre el cual queremos cesar la actividad de obtener la ubicación.

En la primera función de callback, que llamamos cuando todo ha sucedido correctamente, tendremos un objeto al cual podremos acceder a los siguientes métodos, únicamente los 3 primeros estarán accesibles siempre, los otros solo se mostraran si están disponibles:

`coords.latitude` devuelve la latitud como un número decimal

`coords.longitude` devuelve la longitud como un número decimal

`coords.accuracy` devuelve la precisión de la posición como un número entero.

`coords.altitude` devuelve la altitud como un número entero.

`coords.altitudeaccuracy` devuelve la precisión de la altitud como un número entero.

`coords.heading` devuelve los grados que hay desde el norte en sentido a las agujas del reloj.

`coords.speed` devuelve la velocidad en metros por segundo.

`timestamp` devuelve la fecha y la hora en formato de fecha de 13 caracteres.

En caso de producirse algún error, en la segunda función de callback, podremos acceder a los siguientes métodos:

`code` devuelve el código de error que se ha generado en la petición.

`message` devuelve un mensaje de error describiendo los detalles del error encontrado.

`PERMISION_DENIED` se produce cuando no da permiso la API de geo localización para acceder a la posición. El código numérico es el 1.

`POSITION_UNAVIABLE` se produce cuando la posición del usuario no puede ser determinada. El código numérico es el 2.

`TIMEOUT` se produce cuando el tiempo sobrepasa al tiempo estipulado en el método de obtención de la posición. El código numérico es el 3.

`UNKNOWN_ERROR` se produce cuando desconocemos el error que se ha producido. El código numérico es el 0.

Como opciones en la función para obtener la posicion podemos definir los siguientes parámetros:

`máximumAge` se utiliza para especificar que no se aceptara una posición que haya sido detectada a un tiempo anterior al especificado. Este tiempo se especifica en milisegundos.

`timeout` se utiliza para especificar el tiempo de demora máximo que puede tardar la función en ejecutarse. Este tiempo se especifica en milisegundos.

`enableHighAccuracy` se utiliza para exigir una mayor precisión en la toma de medidas, esta opción puede utilizarse para restringir algunos dispositivos que no tienen la precisión necesaria.

2.5.7. Canvas

Esta versión de HTML5 nos proporciona una nueva API denominada canvas. Este elemento tiene la propiedad de albergar gráficos dibujados a través de un lenguaje de scripting (generalmente javascript). El elemento canvas no es mas que el contenedor, los dibujos los realiza el propio script. Una vez el elemento canvas se halla en nuestro documento, podemos empezar a dibujar llamando al método `getContext("2D")`, que devolverá un objeto el cual utilizaremos para definir nuestro dibujo. A continuación veremos todas las propiedades y atributos que podemos utilizar para realizar estos dibujos.

Primeramente veremos las formas que podemos definir en nuestro dibujo:

`rect(y, x, width, height)` se utiliza para definir un rectángulo, donde x e y son la posición que ocupara el rectángulo dentro del elemento canvas en pixeles de la esquina superior izquierda del rectángulo. Width y height son el ancho y la altura que tendrá el rectángulo. Para que este rectángulo se represente visualmente dentro del elemento canvas, debemos llamar posteriormente a los métodos `fill()` o `stroke()`.

`fillRect(x,y,width, height)` se utiliza para definir un rectángulo, los parámetros se utilizan de la misma manera que el caso anterior, pero con la diferencia que este rectángulo se

mostrara coloreado con el color que se haya definido con la propiedad fillStyle, en caso de no haberse definido se mostrara de color negro, que es el color por defecto.

`strokeRect(x, y, width, height)` se utiliza para definir un rectángulo, los parámetros se utilizan de la misma manera que en los casos anteriores, pero con la diferencia que este rectángulo no tiene relleno, solo son las líneas del contorno (perímetro). El color de este rectángulo será el que se haya definido en la propiedad strokeStyle, en caso de no haberse definido se mostrara de color negro, que es el color por defecto. Este elemento se utiliza principalmente para dibujar bordes en los rectángulos que si tienen relleno.

`clearRect(x, y, width, height)` se utiliza para definir un rectángulo para vaciar los pixeles que se hallen en su interior. El efecto es el mismo como si pintáramos un rectángulo blanco. Los parámetros se utilizan de la misma manera que en los casos anteriores.

`fill()` se utiliza para rellenar el elemento que estemos dibujando según el color que se haya definido en la propiedad fillStyle, en caso de no haberse definido se pintara con el color por defecto que es el negro.

`beginPath()` se usa para definir una línea o camino (conjunto de líneas) que será dibujado. Esta línea la definiremos con los parámetros que mostraremos a continuación.

`moveTo(x, y)` se usa para mover el punto de lugar de dibujo de una línea, no dibuja ninguna línea. Los parámetros x e y es la posición según ele eje que cada parámetro representa. Para empezar a dibujar una línea debemos definir donde queremos empezar con este método.

`lineTo(x, y)` se usa para dibujar una línea recta, que ira desde un punto de dibujo definido anteriormente, hasta la posición que indiquemos en los parámetros x e y.

`quadraticCurveTo(cx, cy, x, y)` se utiliza para definir una línea curva, que ira desde un punto de dibujo definido anteriormente hasta la posición que indiquemos en los parámetros x e y. La curva de la línea se hará intentando alcanzar el punto que definamos en cx y cy.

`beizerCurveTo(c1x, c1y, c2x, c2y, x, y)` se utiliza para definir una línea curva, que ira desde un punto de dibujo definido anteriormente hasta la posición que indiquemos en los parámetros x e y. La curva se dibujara intentado alcanzar los dos puntos que definamos en los parámetros c1x, c1y y c2x, c2y consecutivamente.

`arc(x,y,r,anguloinicial,angulofinal,[direccion])` se utiliza para definir una línea en forma de arco, que corresponden al perímetro de un circulo. Los parámetros x e y se usan para posicionar el centro del circulo. El parámetro r es el radio que tendrá este círculo. Los parámetros anguloinicial y angulofinal indican la línea que se dibujara, empezando en el 0 que se sitúa a las 3 en punto si un reloj se tratara. Estos angulos deben indicarse en radianes. [Radianes = (Math.PI / 180) * grados]. El parámetro opcional dirección indica hacia que lado se empezara a dibujar el arco.

`arcTo(x1,y1,x2,y2,r)` se utiliza para definir una línea curva/forma de arco, donde el punto indicado por x1 y y1 será el inicio de esta línea y x2 e y2 será el final. La curvatura del arco se vera definida por el parámetro r que indica el radio de la circunferencia, hay que tener cuidado

al indicar el radio ya que si este es demasiado grande o demasiado pequeño se verán afectados los puntos de inicio y final que definamos.

`closePath()` se utiliza para unir el principio con el final de una línea o conjunto de estas, para crear así una figura cerrada. Si tenemos una figura que no esta cerrada, debemos usar esta propiedad si queremos pintar el relleno con el método `fill()`.

`stroke()` se utiliza para dibujar en el documento las líneas que hayamos definido utilizando los métodos vistos recientemente. Si no se llama a este método no se representara gráficamente ninguna línea.

`isPointInPath(x,y)` se utiliza para saber si el punto que indiquemos en los parámetros x e y se encuentra en la línea o camino que hayamos definido.

`clip()` se utiliza para añadir una forma a otra ya definida anteriormente, esta segunda se sobrepone a la anteriormente definida.

`drawImage(imagen, x, y,[width, height])` se utiliza para definir una imagen dentro del elemento canvas. En el parámetro imagen debemos indicar la imagen que queremos insertar, con los parametros x e y definimos la posición de la imagen y los parametros opcionales widht y height para modificar la anchura y la altura respectivamente.

`fillText(text, x, y [maxWidth])` se usa para dibujar texto. El texto a dibujar lo indicaremos en el parámetro text, x e y será el punto donde empezaremos a escribir y el parámetro opcional maxWidth será el tamaño máximo, en pixeles, que puede ocupar el texto.

`strokeText(text, x, y [maxWidth])` se utiliza para dibujar el contorno del texto indicado en el parámetro text, los parámetros x e y son la posición donde se empezara a dibujar el texto y el parámetro opcional maxWidth son los pixeles máximos que puede ocupar el dibujo de ese texto.

`measureText(text)` se utiliza para obtener el numero de pixeles que ocupa el texto que pasamos como parámetro.

Llegados a este punto veremos como podemos estilizar y agregar efectos de presentación a los elementos que hayamos dibujado anteriormente:

`font` se utiliza para definir las propiedades del texto utilizando la misma sintaxis que en una hoja de estilos. Podemos definir `font-size`, `font-variant`, `font-weight`, `line-height`, `font-family`, `caption`, `icon`, `menu`, `message-box`, `small-caption` y `statys-bar`.

`fontAlign` se utiliza para definir la alineación del texto, podemos definir `center`, `end`, `left`, `right` y `start`.

`textBaseline` se utiliza para definir la alineación vertical del texto, podemos definir `alphabetic`, `bottom`, `hanging`, `ideographic` y `middle`.

`fillStyle` se utiliza para definir el color con el que se rellenara el contenido de un dibujo.

`strokeStyle` se utiliza para definir el color que tendrá el contorno de una forma que hayamos definido anteriormente.

`shadowColor` se utiliza para definir el color que tendrá la sombra que vayamos a definir.

`shadowBlur` se utiliza para definir el grado de desenfoque que tendrá una sombra.

`shadowOffsetX` y `shadowOffsetY` se utiliza para definir el punto superior izquierdo de la sombra, que será donde se posicione esta.

`createLinearGradient(x1, y1, x2, y2)` se utiliza para crear un degradado lineal de color. Esta función devuelve un objeto, el cual puede ser utilizado para rellenar cualquier dibujo. La linealidad del degradado se dirigirá del punto definido por x1 e y1 al punto definido por x2 e y2.

`createRadialGradient(x1, y1, r1, x2, y2, r2)` se utiliza para crear un degradado radial de color. Esta función devuelve un objeto, el cual puede ser utilizado para rellenar cualquier dibujo. La forma radial del color se vera definida por el punto x1, y1, que representa el centro del circulo del principio del degradado, y r1 su radio. Y por el otro punto x2, y2 que representa el final del circulo del degradado, donde r2 es su radio.

`addColorStop(stop, color)` se utiliza para definir el color que utilizaremos para generar los degradados cuando utilizamos los dos métodos anteriormente vistos. Con el primer parametro, que va de 0 a 1 incluyendo decimales, definimos que porcentaje de distancia que usaremos del color que definimos en el segundo parámetro.

`createPattern(imagen, repetición)` se utiliza para crear un patrón de relleno. El parámetro imagen será una imagen video u otro elemento de canvas, y el parámetro repetición será el tipo de repetición que se dibujara, esta puede definirse como repeat, repeat-x, repeat-y y no-repeat.

`lineCap` se utiliza para definir el estilo del principio y final de una línea o camino (conjunto de líneas). Podemos definir butt, que es la opción por defecto, round que redondea la línea y square que forma un cuadrmmado en la esquina de estas líneas.

`miterLimit` se utiliza para limitar la longitud de una esquina dibujada con la propiedad miter definida, ya que si los ángulos de esas líneas son muy similares la esquina puede alargarse mucho.

`lineJoin` se utiliza para definir el estilo que tendrá una esquina formada por dos líneas que se juntan. Podemos definir bevel, que recortara los bordes sobrantes dejando un corte limpio; round que redondeara la esquina; o bien miter que creara una nueva forma de esquina teniendo en cuenta los ángulos de cada línea.

`lineWidth` se utiliza para definir el ancho de la línea o líneas.

`scale(width,height)` se utiliza para redimensionar un elemento dibujado. Con el primer parametro definimos la anchura y con el segundo la altura. Para representar estos valres se utiliza la escala a 1.

`rotate(angulo)` se utiliza para rotar un elemento dibujado, como parámetro introducimos el ángulo de inclinación de la figura, en radianes. Admite valores negativos para invertir el giro.

`translate(x, y)` se utiliza para desplazar elementos según los parámetros introducidos. Los valores de x e y se incrementan las coordenadas de la figura original.

`transform(a, b, c, d, e, f)` se utiliza para realizar una transformación respecto un elemento original. Con los parámetros a y d se escala el dibujo horizontal y verticalmente respectivamente, con b y c se decanta el elemento horizontal y vertical respectivamente y con e y f se mueve el elemento horizontal y verticalmente respectivamente.

`setTransform(a, b, c, d, e ,f)` se utiliza de la misma manera que transform() pero la diferencia es que esta segunda antes establecer los valores indicados en los parámetros borra las transformaciones anteriormente hechas con transform() y únicamente prevalece la que acabamos de indicar.

`createImageData(width, height)` se utiliza para crear un nuevo objeto formado por un conjunto de pixeles negros trasnparentes por defecto, se definen en valor RGBA que por defecto es (0,0,0,0). Con el parámetro width definimos el ancho y con height la altura.

`imageData.width` se utiliza para definir o redefinir el ancho del objeto imagedata.

`imageData.height` se utiliza para definir o redefinir la altura del objeto imagedata.

`imageData.data` se utiliza para acceder a la composición de colores RGBA de cada pixel del objeto creado. Es un vector de 4 elementos en el que accedemos a cada uno de los atributos del color RGBA en ese orden.

`getImageData(x, y, width, height)` se utiliza para realizar una copia de una parte del dibujo hecho en el elemento canvas, que empezara en el punto x e y que se extenderá tanto como indiquemos en los valore width y height (ancho y alto).

`putImageData(imageData, x, y)` se utiliza para introducir un elemento imageData que pasamos por parámetro dentro del elemento canvas, concretamente en el punto definido por los parámetros x e y.

`globalAlpha` se utiliza para definir la transparencia de todos los elementos que contiene el elemento canvas. El valor va desde 0 a 1 representando el porcentaje de opacidad.

`globalCompositeOperation` se utiliza para definir como se posicionan las formas dentro del elemento canvas. Esta propiedad se refiere a la superposición entre estos. Podemos definir `source-atop`, `source-in`, `source-out`, `source-over`,`destination-atop`, `destination-in`, `destination-out`, `destination-over`, `lighter`, `copy` y `xor`.

`save()` y `resotore()` se utiliza para guardar (save) estados de una forma y después poder recuperarlos (restore). Esta propiedad tiene importancia cuando realizamos varios cambios a una misma figura y si en algún momento nos interesa volver a la figura original podemos hacerlo fácilmente con estos dos métodos.

2.5.8. Gráficos vectoriales (SVG API)

En la quinta versión de HTML disponemos de una manera alternativa a canvas para dibujar sobre el documento web. Esta otra manera tiene la cualidad de poder ser cargada desde un fichero externo, y no es necesario tenerlo en el propio documento. Pero también se puede hacer del modo tradicional y definirla en el propio documento. A continuación veremos como podemos realizar dibujos con esta otra herramienta. Este documento utiliza la misma nomenclatura que html o xml, donde los distinos elementos se encierran entre las claves < y >.

Primero de todo, veremos las formas que podemos realizar con esta nueva API.

`rect` se utiliza para definir rectángulos. Para definir el rectángulo debemos añadir los atributos `width` y `height` para concretar la anchura y la altura. También podemos definir los atributos `x` e `y` que definirán el margen izquierdo y superior respectivamente. Además también podemos definir los atributos `rx` y `ry` para definir el redondeado las esquinas, de esta manera todas las esquinas quedan definidas de la misma manera.

`circle` se utiliza para definir un círculo. Añadiendo los atributos `cx` y `cy` definiremos el punto que será el centro del círculo, si no lo hacemos, por defecto ese punto será el (0,0). También debemos definir un atributo `r` que determinara la longitud del radio.

`ellipse` se utiliza para definir una elipse. Si añadimos los atributos `cx` y `cy` definiremos el punto que será el centro de la elipse. También podemos añadir los atributos `rx` y `ry` que serán con los que definiremos la elipse, con `rx` definimos el ancho y con `ry` la altura.

`line` se utiliza para definir una línea recta. Con los atributos `x1` e `y1` definiremos el punto de partida de la línea, y con los parámetros `x2` e `y2` definiremos el punto final de esta línea.

`polyline` se utiliza para definir un conjunto de líneas rectas. Debemos incluir el atributo `points` en su declaración, y este debe contener parejas de puntos separados por una coma, y por un espacio en banco entre cada punto que serán por todos los que pasaran las líneas.

`polygon` se utiliza para definir polígonos. Debemos definir el atributo `points` de la misma manera que el caso anterior solo que esta vez estos puntos representaran los vértices del polígono que definamos.

`path` se utiliza para definir un camino o conjunto de líneas. Para definir este elemento debemos insertar el atributo `d` seguido de las coordenadas. Para indicar el propósito de nuestra acción debemos introducir la letra identificador de cada tipo de acción que queremos realizar seguida de las dos coordenadas.

Letra	Equivalente en canvas
M	MoveTo
L	LineTo
H	LineTo horizontal
V	LineTo Vertical
C	quadraticCurveTo
S	quadraticCurveTo suavizada
Q	beizerCurveTo
T	beizerCurveTo suavizada
A	ArcTo
Z	closePath

Figura2_79: Tabla de identificadores de línea del atributo d.

```
d="M150 0 L75 200 L225 200 Z"
```

Figura2_80: Ejemplo de código para definir un triangulo con el atributo d.

`text` se utiliza para dibujar el texto que contiene este elemento. Con los parámetros `x` e `y` definimos el punto donde empezara a dibujarse el texto. Dentro del mismo elemento podemos definir múltiples líneas con el elemento `tspan`, al cual le debemos redefinir los atributos `x` e `y` para que no se solapen uno encima del otro.

Si queremos cambiar el estilo de nuestra figura, también podemos hacerlo definiendo el atributo style cuando declaramos cualquier elemento de los vistos anteriormente. La nomenclatura es la misma que en CSS, es mas podemos definir el estilo con las propiedades de CSS. Además SVG dispone de una serie de atributos más adecuadas desenvolvernos mejor en este entorno, a continuación veremos algunas de las más destacadas:

`opacity` se utiliza para definir la transparencia del elemento que estamos definiendo.

`fill` se utiliza para definir el color de relleno del elemento que estamos definiendo.

`fill-opacity` se utiliza para definir la transparencia del color que hemos definido para el rellenado.

`stroke` se utiliza para definir el color del contorno del elemento que estamos definiendo.

`stroke-width` se utiliza para definir el ancho del contorno del elemento que estamos definiendo.

Para ir un poco más allá en el estilizado, podemos añadir diferentes efectos a nuestro dibujo. Para utilizar estos efectos debemos declarar un elemento a parte llamado `defs` el cual contenga cada uno de los diferentes efectos que vayamos a utilizar. A cada uno de estos elementos de le designaremos un identificador. Una vez declarado nuestro efecto, para poder utilizarlo basta con introducir como valor donde lo queramos utilizarlo `url(#identificadordelefecto)`.

SVG nos proporciona un gran abanico de efectos aplicables, los efectos disponibles son `feDistantLight`, `fePointLight`, `feSpotLight`, `feBlend`, `feColorMatrix`, `feComponentTransfer`, `feComposite`, `feConvolveMatrix`, `feDiffuseLighting`, `feDisplacementMap`, `feFlood`, `feGaussianBlur`, `feImage`, `feMerge`, `feMorphology`, `feOffset`, `feSpecularLighting`, `feTitle` y `feTurbulence`. Además también disponemos de los degradados lineales y radiales que también pueden ser definidos dentro de la etiqueta `defs` y utilizarlos como un efecto mas.

2.5.9. Trabajadores en segundo plano (Web workers API)

Esta API de la nueva versión de HTML, nos ofrece la posibilidad de ejecutar código javascript en segundo plano. Anteriormente, en pasadas ediciones, cuando el código javascript era ejecutado el navegador únicamente realizaba esa tarea, dejando en un segundo plano todas las demás funcionalidades. Utilizando esta API, podemos permitir al usuario navegar por la web mientras se ejecuta un código javascript muy pesado, que requiere de muchos segundos de cálculo, o bien podemos tener código javascript ejecutándose continuamente sin que se vea afectado el navegador.

Antes de todo, para utilizar un web worker debemos tener creado un fichero con el código javascript en el (archivo con extensión .js).

Para usar un web worker en nuestro documento debemos crear un nuevo objeto pasando como parámetro nuestro fichero de javascript creado anteriormente:

```
var worker = new Worker('nuestrofichero.js');
```

Figura2_81: Ejemplo de código de la inicialización de un web worker.

Si queremos hacer que nuestro worker pueda ser compartido debemos hacerlo de la siguiente manera:

```
var worker = new SharedWorker('worker.js');
```

Figura2_82: Ejemplo de código de la inicialización de un web worker compartido.

Para establecer una comunicación entre el fichero javascript y nuestro documento debemos utilizar el método `postMessage(datos)`; Con este método enviamos al otro documento los datos que nosotros necesitemos. Para recibir los datos tenemos los eventos `onmessage` y `onerror` (o bien `message` y `error` para usarlos en un event listener), con los que accederemos a los datos de ese evento. Para ello basta con utilizar `event.data` para acceder a los datos.

Finalmente, para finalizar un web worker y liberar así a nuestro navegador debemos llamar al método `terminate()`.

```
worker.terminate();
```

Figura2_83: Ejemplo de código de la finalización de un web worker.

2.5.10. Mensajes (Messaging API)

La nueva versión de HTML ofrece una API con la posibilidad de enviar mensajes entre documentos web, utilizando la URL de cada documento como identificador.

Para mandar un mensaje debemos utilizar el método `window.postmessage(mensaje, urldestinatario)`, el primer parámetro es el mensaje en cuestión, que debe ser un formato de texto, y el segundo debe ser la URL del servidor destinatario.

En el documento receptor, para recibir el mensaje debemos utilizar el listener de eventos y capturar el evento `message` . Así podemos capturar los mensajes entrantes y mediante la función de callback obtener el mensaje. Para obtener el mensaje debemos acceder al `event.data`. También podemos comprobar la procedencia del mensaje (URL) mediante `event.origin`.

En caso de producirse algún error, podemos capturarlo mediante `window.onerror`, que obteniendo el error podemos conseguir mas información del problema sucedido. Los errores posibles son: `UNKNOWN_ERROR`, `INVALID_ARGUMENT_ERROR`, `TIMEOUT_ERROR`, `PENDING_OPERATION_ERROR`, `IO_ERROR`, `NOT_SUPPORTED_ERROR`, `PERMISSION_DENIED_ERROR` y `MESSAGE_SIZE_EXCEEDED`.

2.5.11. Comunicación socket (Web Sockets API)

La versión cinco de HTML nos proporciona una API que nos permite establecer conexiones bidireccionales entre cliente y servidor que pueden ser abiertas desde el código javascript. Estas conexiones permiten el flujo de texto entre cliente y servidor.

Para establecer una conexión con el servidor debemos crear un nuevo objeto WebSocket pasando como parámetro la URL:

```
var ws = new WebSocket("ws://miservidor.com")
```

Figura2_83: Ejemplo de código de inicialización de un socket.

Una vez hecho esto, tenemos a nuestra disposición cuatro eventos para controlar la conexión con el servidor:

`open` se lanza cuando la conexión de socket se ha establecido.

`message` se lanza cuando se recibe un mensaje del servidor.

`error` se lanza cuando ocurre cualquier error en la comunicación.

`close` se lanza cuando la conexión de socket se cierra.

Para poder acceder al mensaje recibido debemos utilizar `event.data`, de este modo accederemos al mensaje que debe ser de tipo texto.

Para comunicarnos con el servidor, disponemos de el método `ws.send(mensaje)` que nos permitirá enviar un mensaje al servidor.

Para finalizar la conexión, debemos emplear el método `ws.close()`, de esta manera se cerrara la conexión establecida entre el servidor y el cliente.

2.5.12. Almacenamiento local (Local Storage API)

HTML5 nos proporciona una API que nos ofrece la posibilidad de almacenar datos en nuestro sistema de manera permanente y poder acceder a ellos cuando lo precise nuestra aplicación web. Esta API nos permite almacenar datos de las siguientes formas:

Almacenamiento local

Gracias a esta API, se nos brinda la posibilidad de almacenar datos en nuestro navegador. Para ello tenemos a nuestra disposición las los objetos `localStorage` y `sessionStorage` sobre los que podemos almacenar los datos. La diferencia entre los dos esta en que el almacenamiento local se almacena de manera persistente el navegador, de manera global. En cambio el almacenamiento de sesión se almacena únicamente para esa sesión, en esa pestaña concreta.

Para utilizar la API disponemos de los siguientes métodos que podemos utilizar en los objetos localStorage y sessionStorage, para ambos son los mismos:

`setItem(nombre,datos)` se utiliza para almacenar los datos, el primer parámetro será el nombre de que tendrá ese dato y el segundo parámetro serán los datos en si. Vendría a ser el nombre de una variable y su contenido.

`getItem(nombre)` se utiliza para obtener un elemento almacenado, pasando como parámetro su nombre.

`removeItem(nombre)` se utiliza para borrar un elemento almacenado, pasando como parámetro su nombre.

`clear()` se utiliza para vaciar todos los elementos guardados en el objeto de almacenamiento.

Bases de datos

Esta API de almacenamiento nos permite crear bases de datos para utilizar en nuestras aplicaciones web. Para crear una nueva base de datos debemos utilizar el siguiente comando:

```
openDatabase(“Nombre de la base de datos”, “numero de versión de la base de datos”, “Descripción de la base de datos”, tamaño en bytes de la base de datos);
```

Figura2_84: Ejemplo de código de la inicialización de una base de datos utilizando la API de almacenamiento.

Este método nos devolverá un objeto que será nuestra base de datos. El tamaño indicado puede ser variable, pero es para que el navegador prepare esa cantidad de bytes desde un principio. Si no se puede crear la base de datos por cualquier razón el método devolverá el valor NULL.

Para realizar una transacción sobre la base de datos debemos utilizar uno de los siguientes métodos del objeto devuelto anteriormente:

```
readTransaction(callback_function(tx))
```

Figura2_85: Ejemplo de código para realizar una transacción de lectura sobre la base de datos.

```
Transaction(callback_function(tx))
```

Figura2_86: Ejemplo de código para realizar una transacción sobre la base de datos.

El primer método no permite la modificación de la base de datos, y para controlar mejor la aplicación se recomienda utilizar este tipo para realizar consultas de solo lectura. Sobre el objeto de la transacción `tx` podemos ejecutar las consultas con la siguiente sentencia:

```
ExecuteSQL('consultaSQL', [variables externas que usamos en la consulta],
funcion_exito(tx, result), función_error(tx, error));
```

Figura2_87: Ejemplo de código para realizar una consulta sobre la base de datos.

El primer parámetro es la consulta que queremos realizar sobre la base de datos en lenguaje SQL, es el único parámetro obligatorio, todos los demás son opcionales. El segundo parámetro son variables externas que podemos utilizar en la consulta. El tercer parámetro es la función de callback que se llamara si la ejecución de la consulta se ha realizado correctamente. Y finalmente el último parámetro es la función de callback que se realizara si surge algún error.

En caso de haberse llamado a la función de éxito, del objeto `result` podemos utilizar los siguientes métodos:

`insertedId` se utiliza para obtener el identificador de la fila añadida, únicamente en el caso de realizar un Insert. En caso diferente se lanzara una excepción.

`rowsAffected` se utiliza para obtener el numero de filas afectadas por un cambio en la base de datos. Si no se realizan cambios, se devuelve un 0.

`Rows[n]` se utiliza para acceder a las filas del resultado de la consulta. Cada valor de n pertenece a una fila.

`item(n)` se utiliza para acceder a un elemento de una fila. Cada valor de n representa un elemento de la fila.

`length` se utiliza para devolver el número de filas obtenidas.

En caso de haberse llamado a la función de error, estos son los métodos que podemos utilizar sobre el objeto error:

`code` se utiliza para obtener el código de error

`message` se utiliza para obtener el mensaje de error.

Evento storage

En los dos casos vistos anteriormente, podemos utilizar la aplicación simultáneamente en varias pestañas de nuestro navegador por ejemplo, utilizando la misma base de datos o el mismo elemento de almacenamiento local. Esto no supone ningún problema de estabilidad, pero si un problema para el usuario ya que no se percata de los cambios realizados por otro usuario u otra aplicación ejecutándose simultáneamente.

Para solucionar este problema, la API de almacenamiento local nos proporciona el evento `storage` que nos permite captar los cambios en los medios de almacenamiento, ya sean por una adición, borrado o modificación. Una vez capturado el evento, podemos utilizar los siguientes métodos sobre el:

`key` se utiliza para obtener el identificador único de un elemento cambiado, en caso de adición se recibirá una cadena vacía.

`oldvalue` se utiliza para obtener el valor que contenía un elemento cambiado, en caso de adición se recibirá una cadena vacía.

`newvalue` se utiliza para obtener el valor que contendrá un elemento cambiado.

`url` se utiliza para obtener la dirección url del documento que ha realizado la modificación.

`storageArea` se utiliza para obtener los elementos que se han visto afectados por el cambio.

Offline Application

Para utilizar nuestra aplicación offline, es decir, desconectados de la red, debemos crear un fichero con la extensión .manifest en este fichero debemos alojar los siguientes datos:

`CACHE MANIFEST`

`CACHE:`

Ahora debemos poner una lista con todos los archivos que utilizara nuestro documento web de manera offline. También debemos incluir el propio documento.

`FALLBACK:`

A continuación debemos poner el nombre del archivo que usábamos de manera online y que ahora ya no utilizaremos, emplearemos otros en su lugar. Para realizar esto demo poner el

archivo que utilizábamos en modo online, un espacio y el nombre del archivo que usaremos en modo offline.

Y finalmente si tenemos archivos que únicamente pueden ser utilizar de manera online, debemos agregar lo siguiente al texto:

`NETWORK:`

Finalmente, en esta sección, indicaremos la lista de los archivos que únicamente se pueden utilizar de manera online.

Una vez creado este archivo, dentro de la etiqueta html de nuestro documento debemos agregar nuestro archivo manifest:

```
<!DOCTYPE HTML>
<html lang=”es” manifest=”nombre_del_archivo.manifest”>
    <head>
        <meta charset="UTF-8">
        <title>Titulo del documento</title>
```

Figura2_88: Ejemplo de código de la inclusión del archivo manifest dentro de la etiqueta `html`.

CAPITULO 3: CSS3

En este capitulo veremos el nuevo estándar de hojas de estilo. Podemos ver como funcionan estos efectos podemos consultar las webs http://www.w3schools.com/css3/default.asp y http://www.css3.info/preview/. En estos enlaces (y en otros que podríamos facilitar) nos proporcionan una colección de ejemplos que nos ilustran de las nuevas características que CSS3 nos aporta que a continuación vamos a detallar.

3.1. Estándar CSS

Las CSS, de sus siglas en ingles Cascading Style Sheets, son un lenguaje que se utiliza para definir el estilo de presentación que tendrá un documento HTML o XML y por extensión XHTML. Estas hojas de estilo se utilizan para separar la estructuración del documento de la presentación del mismo. Constituyen por tanto el complemento ideal para todo lo visto en el capítulo anterior. En aquel se proporcionan los mecanismos para estructurar la información en formato web. En este capítulo trataremos todo lo relacionado con la presentación de la misma.

Al ser un lenguaje de tipo texto, al igual que HTML, también podemos crearlo y editarlo utilizando cualquier procesador de texto o bien usando editores más especializados como DreamWeaver, WebStorm o WebMatrix, por poner algunos ejemplos.

Su uso es bastante sencillo, consiste, primeramente en seleccionar los elementos que queremos estilizar, donde podemos emplear los selectores para ayudarnos en nuestra selección. Una vez tenemos el elemento o elementos seleccionados, en el interior de las llaves { y } debemos definir, mediante una serie de palabras clave, las propiedades que tendrá la presentación de ese elemento que seleccionemos.

3.2. El nuevo estándar CSS3

CSS3 es la nueva versión de CSS que viene con múltiples mejoras y nuevas características. A diferencia de la HTML5, CSS no elimina ni modifica la especificación de las antiguas versiones, de modo que todo lo ya existente sigue siendo es perfectamente valido en esta nueva versión. CSS3 trae mejoras en aquellos aspectos que CSS2 no cubría o que le faltaba utilidad, proporcionando así un mayor control al desarrollador sobre la presentación. De este modo, características que antes no podíamos implementar directamente, ya que debíamos acudir a lenguajes externos como javascript, ahora definiendo una única característica con la que podemos obtener los mismos resultados. Evidentemente CSS3 no ampara todas las presentaciones de elementos que uno pueda imaginar, pero si abarca buena parte de estas.

3.3. Selectores de CSS3 (Selectors)

Los selectores de CSS3 se utilizan para poder seleccionar un determinado elemento del documento HTML para poder así formatearlo a nuestro agrado. Las anteriores versiones de CSS ya nos ofrecían diferentes maneras de hacerlo, pero en esta nueva versión se han proporcionado otras nuevas maneras para seleccionar elementos del documento. A continuación vamos a ver los nuevos selectores que se han implementado en la nueva versión de CSS3. Para que todos estos selectores funcionen debe ser declarado <!DOCTYPE> en el documento.

`elemento1~elemento2` sirve para seleccionar el elemento2 cuando esta precedido por el elemento1, estos dos elementos deben pertenecer al mismo padre y no tienen por qué estar consecutivamente.

```
h2 ~ p
```

Figura3_01: Ejemplo de código utilizando el selector ~.

En la figura3_01 seleccionamos un texto p únicamente cuando esta detrás de un titulo 2.

`:checked` sirve para seleccionar un elemento checkeado, esta opción solo puede aplicarse a los checkboxes y a los radiobuttons.

```
input:checked
```

Figura3_02: Ejemplo de código utilizando el selector :checked.

En la figura3_02 seleccionamos un input checkbox o radiobutton cuando es marcado o seleccionado respectivamente.

`:disabled` sirve para seleccionar los elementos que tienen el atributo disabled activado.

```
input:disabled
```

Figura3_03 Ejemplo de código utilizando el selector :disabled.

En la figura3_03 seleccionamos los elementos del input, los cuales están en disabled.

`:empty` sirve para seleccionar aquellos elementos de nuestro documento que no tengan hijos, para los elementos de texto p también es valido.

```
article:empty
```

Figura3_04: Ejemplo de código utilizando el selector :empty.

En la figura3_04 seleccionamos los elementos article los cuales no contienen nada en su interior.

`:enabled` sirve para seleccionar aquellos elementos los cuales tienen el atributo enabled activado.

```
input:enabled
```

Figura3_05: Ejemplo de código utilizando el selector :enabled.

En la figura3_05 estamos seleccionando un input cuyo atributo enabled esta activado.

`:first-of-type` sirve para seleccionar el primer elemento del tipo que nosotros indicamos del mismo padre.

```
li:first-of-type
```

Figura3_06: Ejemplo de código utilizando el selector :first-of-type.

En la figura3_06 seleccionamos el primer elemento de una lista.

`:last-child` sirve para seleccionar cualquier elemento, el cual sea el ultimo del propio padre.

```
img:last-child
```

Figura3_07: Ejemplo de código utilizando el selector :last-child.

En la figura3_07 seleccionamos una imagen, la cual es el último elemento perteneciente a ese padre.

`:last-of-type` sirve para seleccionar el ultimo elemento del tipo que nosotros indicamos pertenecientes al mismo padre.

```
p:last-of-type
```

Figura3_08: Ejemplo de código utilizando el selector :last-of-type.

En la figura3_08 estamos seleccionando el último párrafo de un determinado elemento.

`:not(selector)` sirve para seleccionar todos los elementos a excepción del que indicamos nosotros.

```
:not(h1)
```

Figura3_09: Ejemplo de código utilizando el selector :not(selector).

En la figura3_09 seleccionamos todos los elementos a excepción de h1.

`:nth-child(n)` sirve para seleccionar el elemento que esta en n posición dentro del mismo padre. La posición es independiente al tipo de cada elemento. n puede ser un numero una palabra clave o una formula.

```
p:nth-child(3)
```

Figura3_10: Ejemplo de código utilizando el selector :nth-child(n).

En la figura3_10 estamos seleccionando el tercer elemento, únicamente si ese elemento es de tipo p.

`:nth-last-child(n)` funciona igual que el elemento anterior, la única diferencia es que empieza a contar por el ultimo elemento en vez del primero.

```
img:nth-last-child(2)
```

Figura3_11: Ejemplo de código utilizando el selector :nth-last-child(n).

En la figura3_11 seleccionamos la imagen que se posiciona como elemento en penúltimo lugar de ese padre.

`:only-child` sirve para seleccionar los elementos que son hijos únicos.

```
p:only-child
```

Figura3_12: Ejemplo de código utilizando el selector :only-child.

En la figura3_12 seleccionamos aquellos párrafos que son elementos únicos dentro de ese padre.

`:root` sirve para seleccionar el elemento raíz del documento. En los documentos html el elemento raíz siempre es el elemento html.

```
:root
```

Figura3_13: Ejemplo de código utilizando el selector :root.

En la figura3_13 seleccionamos el elemento raíz del documento, que al ser un documento html es el elemento con el mismo nombre.

`:target` sirve para seleccionar los elementos destino de un punto de anclaje dentro de nuestro documento. Se deseleccionan utilizando otro punto de anclaje.

```
article:target
```

Figura3_14: Ejemplo de código utilizando el selector :target.

En la figura3_14 estamos seleccionando un elemento article después de haber sido alcanzado un punto de anclaje.

`::selection` sirve para seleccionar la parte que es seleccionada por el usuario con el raton. Con este selector solo podemos modificar color, background, cursor, y outline.

```
p::selection
```

Figura3_15: Ejemplo de código utilizando el selector ::selection.

En la figura3_15 estamosseleccionando la parte que el usuario haya seleccionado de ese párrafo.

`[atributo^=valor]` sirve para seleccionar aquellos elementos cuyo valor del atributo elegido empiece por el valor que nosotros le indicamos.

```
a[href^=”http”]
```

Figura3_16: Ejemplo de código utilizando el selector [atributo^=valor].

En la figura3_16 seleccionamos todos los enlaces cuya referencia sea una dirección que empiece por la cadena “http”, es decir, un enlace externo al propio documento.

`[atributo$=valor]` sirve para seleccionar aquellos elementos cuyo valor del atributo elegido sea exactamente el mismo valor que nosotros indicamos.

```
article[id$=”extra”]
```

Figura3_17: Ejemplo de código utilizando el selector [atributo$=valor].

En la figura3_17 seleccionamos los elementos article cuyo identificador sea “extra” exactamente.

`[atributo*=valor]` sirve para seleccionar aquellos elementos cuyo valor del atributo elegido contenga el valor que nosotros indicamos.

```
a[href*=”.pdf”]
```

Figura3_18: Ejemplo de código utilizando el selector [atributo*=valor].

En la figura3_18 seleccionamos aquellos elementos cuya referencia contenga “.pdf”, seleccionando de nuestro documento aquellos enlaces que sean pdfs.

3.4. Bordes (Borders)

En esta nueva versión de CSS3 se han añadido nuevas propiedades referentes a los bordes.

`border-radius` se utiliza para redondear las esquinas de los bordes de las cajas. Pueden ser redondeadas todas por igual o independientemente con diferentes valores. Tambien podemos definir los bordes individualmente mediante `border-bottom-left-radius`, `border-bottom-right-radius`, `border-top-left-radius` y `border-top-right-radius`.

`box-shadow` se utiliza para crear un efecto de sombreado de los distintos elementos. Se puede elegir la posición de esta, así como su color.

`border-image` se utiliza para emplear una imagen como borde, sin necesidad de utilizar ningún programa de retocado de imágenes. Ajustando las propiedades podemos utilizar la imagen sin necesidad de retocarla, será el propio navegador el que se encargue de redimensionar la imagen para que se ajuste a los valores indicados.

3.5. Fondo (Background)

A continuación veremos las nuevas propiedades que se han añadido a CSS3 en relación con los backgrounds o el fondo de un elemento.

`background-size` se utiliza para definir las medidas de la imagen de fondo desde la misma hoja de estilos. Estas medidas pueden ser tanto en pixeles como en porcentajes.

`background-origin` se utiliza para especificar el área sobre la cual tendrá repercusión el fondo. Puede definirse las áreas para `content-box`, `padding-box` o `border-box`.

`background-clip` se utiliza para delimitar el área a la cual afectara las propiedades de coloreado. Al igual que el caso anterior puede especificarse para `content-box`, `padding-box` o `border-box`.

En css3 podemos definir múltiples imágenes de fondo si separamos sus rutas por una coma.

3.6. Efectos del texto (Text effects)

A continuación veremos las propiedades que se han añadido a CSS3 en relación con estilar el texto.

`hanging-punctuation` se utiliza para permitir o no el uso de signos de puntuación al inicio o final de texto. Esto tiene sentido bloquearlo si queremos que nuestro texto tenga una determinada forma que no se vea afectada por insertar ningún signo de puntuación.

`punctuation-trim` se utiliza para determinar si debe ser cortado un determinado signo de puntuación según donde este posicionado, o si se encuentra junto a otro de forma contigua.

`text-align-last` se utiliza para definir la alineación de la ultima línea de un párrafo o un bloque de texto cuando el estilo de este es justificado.

`text-emphasis` se utiliza para marcar un texto que queremos dotar de mas énfasis. Están definidos los valores `text-emphasis-style` y `text-emphasis-color`.

`text-justify` se utiliza para especificar como será el justificado en el caso que la alineación del texto sea tal. Estos valores pueden ser `auto`, `inter-word`, `inter-ideograph`, `inter-cluster`, `distribute`, `kashida` o `none`.

`text-outline` se utiliza para especificar el perfil del texto, cuando lo definimos debemos especificar el grosor de este, así como el color. De manera opcional podemos elegir el enfoque.

`text-overflow` se utiliza para especificar que hacer cuando un texto no cabe en el área que tiene estipulada para que este no desborde. Podemos definir que lo corte sin más, que se añadan puntos suspensivos o bien indicar un texto a mostrar.

`text-shadow` se utiliza para dar el efecto de sombra al texto, podemos definir la posición, la profundidad y el color de esta.

`text-wrap` se utiliza para definir que ocurre cuando una frase que no entra en el área de texto, podemos definir que la rompa normal, que no la rompa, que la rompa por donde quiera o dejarlo en manos del navegador.

`word-break` se utiliza para definir como podemos romper una palabra para realizar un salto de línea, podemos definir normal, que se rompa por cualquier sitio o bien que intente conservar las palabras completas.

`word-wrap` se utiliza para romper una palabra cuando no cabe en el área de texto, evitando así que salga. Si ponemos esta propiedad el valor `normal` se aplican las leyes normales, pero si el valor es `break-word` fuerza a que se rompa la palabra por donde sea para que no sobresalga del área estipulada de texto.

3.7. Fuentes (Fonts)

En versiones pasadas de CSS únicamente se podían utilizar fuentes que estuvieran instaladas en el sistema del usuario, en CSS3 podemos utilizar fuentes que son descargadas por el navegador para poder visualizar la web correctamente.

`@font-face` se usa para definir la nueva fuente que emplearemos. Debemos definir la Font-family y la url donde este ubicada la fuente para ser descargada por el usuario. También podemos definir el alargamiento, el estiramiento, la anchura y el rango Unicode.

3.8. Transformaciones (Transforms)

A continuación veremos las nuevas propiedades de CSS3 en lo referente a las transformaciones de elementos tanto en 2D como en 3D.

`transform: translate(x,y)` se utiliza para desplazar el elemento 2D a través del documento tomando como referencia el origen de la posición original. También podemos utilizar `transtaleX(n)` y `translateY(n)`, funcionan de la misma manera, pero estas solo afectan al eje al cual hacen referencia. En el caso de ser elementos 3D, podemos utilizar `translateZ(n)` para el eje de z.

`transform: translate3d(x,y,z)` se utiliza para desplazar el elemento 3D a través del documento tomando como referencia el origen de la posición original.

`transform: rotate(grados)` se utiliza para rotar en sentido de las agujas del reloj un elemento indicando los grados de inclinación. Si estos grados son negativos se invierte el giro, realizándose en sentido opuesto.

`transform: scale(x,y)` se utiliza para escalar (cambiar de tamaño) un elemento 2D. Podemos indicar el ratio de modificación tanto para el eje x como del eje y por separado. También podemos utilizar `scaleX(n)` y `scaleY(n)`, funcionan de la misma manera, pero estas solo afectan al eje al cual hacen referencia. En el caso de ser elementos 3D, podemos utilizar `scaleZ(n)` para el eje de z.

`transform: scale3d(x,y,z)` se utiliza para escalar (cambiar de tamaño) un elemento 3D. Podemos indicar el ratio de modificación de los tres ejes por separado.

`transform: skew(x,y)` se utiliza para inclinar un determinado elemento según los grados que indiquemos, podemos indicar la inclinación tanto en el eje de x como en el de y; también, si introducimos un ángulo negativo invertimos la inclinación hacia el lado opuesto. También podemos utilizar `skewX(n)` y `skewY(n)`, funcionan de la misma manera, pero estas solo afectan al eje al cual hacen referencia.

`transform: matrix(n,n,n,n,n,n)` se utiliza para combinar todos los efectos referentes a los elementos 2D, es una matriz de 6 parámetros para definir los efectos en un solo lugar.

`transform: matrix3d(n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n)` se utiliza para combinar todos los efectos referentes a elementos 3D, es una matriz de 16 parámetros para definir todos los efectos en un solo lugar.

`transform-origin(x,y,(z))` se utiliza para cambiar la posición original de un elemento en el documento. Podemos definir el eje de x y el eje de y. Esta propiedad también es valida para elementos 3D, así que si es el caso, podemos definir también el eje z.

`transform: rotate3d(x,y,z)` se utiliza para rotar un elemento 3d según como definamos cada parámetro para cada eje de rotación. También podemos utilizar `rotateX(n)`, `rotateY(n)` y `rotateZ(n)` para definir la rotación de un único eje.

`perspective(n)` se utiliza para cambiar la perspectiva(inclinación) de un elemento 3D.

`perspective-origin(x,y)` se utiliza para mejorar la propiedad anterior de perspectiva, pudiendo definir los ejes x e y sobre los cuales se producirá tal efecto.

`transform-style` se utiliza para indicar si debemos proteger las posiciones en 3D que ocupan los elementos hijos o no cuando se ven afectados por otros efectos de un elemento padre. Para definir que se deben conservar utilizamos `preserve-3d` y para indicar lo contrario `flat`.

`backface-visibility` se utiliza para indicar si queremos o no que se vea la parte trasera o la parte inversa de un elemento cuando gira completamente sobre si mismo, proporcionando el contenido al revés. Para activar o desactivar esta característica debemos usar los valores `visible` o `hidden` respectivamente.

3.9. Transiciones (Transitions)

En esta nueva versión de CSS3 podemos definir las transiciones cuando hay un cambio de estilo. A continuación vemos como debemos utilizarlo:

`transition` se utiliza para añadir una transición de un determinado elemento, debemos indicar la propiedad a la cual queremos aplicar la transición acompañada del tiempo en segundos que durara esta. Si queremos definir múltiples transiciones podemos añadir mas separadas por comas.

`transition-property` se utiliza para indicar sobre que elemento o elementos afectara la transición definida. Para que la transición tenga efecto debe especificarse también `transition-duration`.

`transition-duration` se utiliza para indicar el tiempo, en segundos, que tardara la transición del elemento o elementos definidos con `transition-property`.

`transition-timing-function` se utiliza para definir la velocidad o aceleración/deceleración del efecto de la transición. Podemos definir que sea `linear`, `ease`, `ease-in`, `ease-out`, `ease-in-out` o definir una matriz con `cubic-bezier(n,n,n,n)` donde n son valores del 0 al 1.

`transition-delay` se utiliza para definir el tiempo, en segundos, de retardo que tardara el efecto en empezarse a reproducir.

3.10. Animaciones (Animations)

En CSS3 podemos crear animaciones en nuestro documento web sin necesidad de utilizar gifs, flash ni javascript. A continuación explicaremos como podemos hacerlo.

`@keyframes` se utiliza para determinar una animación que tendrá lugar en el cambio entre dos o mas estilos que definamos en el propio CSS. Podemos definir la animación en porcentaje de esta, o bien utilizar las palabras clave `from` y `to` que corresponden al 0% y al 100% respectivamente.

`animation` se utiliza para definir una animación donde debemos incluir 6 propiedades: nombre, duración, tiempo de función, retraso, contador de iterador y dirección. Estas propiedades pueden definirse por separado como vemos a continuación.

`animation-name` se utiliza para especificar el nombre de esa animación.

`animation-duration` se utiliza para especificar el tiempo, en segundos, que tardara la animación en reproducirse completamente.

`animation-timing-function` se utiliza para especificar la aceleración/deceleración con la que se representara nuestra animación. Igual que `transition-timing-function` la temporización de la animación la podemos elegir entre `linear`, `ease`, `ease-in`, `ease-out`, `ease-in-out` y `cubic-bezier(n,n,n,n)` matriz donde n son valores que podemos elegir nosotros mimos.

`animation-delay` se utiliza para especificar, en segundos, el retardo que tendrá la animación en iniciarse. Podemos definir valores negativos, con lo que adelantamos la secuencia de la animación en el tiempo que indiquemos.

`animation-interation-count` se utiliza para definir cuantas veces se repetirá la animación que hayamos definido. Para poder definir infinitas repeticiones utilizamos la palabra `infinite`.

`animation-direction` se utiliza para definir que la reproducción de la animación se produzca en orden inverso. Con la palabra `alternate` las veces impares se reproducirá normal, y las veces pares ser reproducirán en orden inverso.

`animation-play-state` se utiliza para parar o continuar la reproducción de la animación. Esta propiedad tiene más sentido en un contexto donde la animación la podemos parar y reanudar mediante javascript, por ejemplo.

3.11. Multiples columnas (Multiple column)

En esta nueva versión de CSS nos permite mostrar un texto por columnas, sin necesidad de alterar el código hmtl. A continuación veremos como podemos hacerlo.

`columns` se utiliza para definir como se estructuraran las columnas, debemos indicar el acho de estas y la cantidad.

`column-count` se utiliza para definir el número de columnas en las cuales se estructurara el elemento seleccionado.

`column-width` se utiliza para definir la anchura que tendrán las columnas que hemos definido con `column-count`.

`column-fill` se utiliza para definir como se rellenaran las columnas, si estarán compensadas intentando igualar el contenido en todas ellas o por el contrario se ignorara este hecho.

`column-gap` se utiliza para definir la distancia de separación que habrá entre las columnas.

`column-rule` se utiliza para definir el aspecto que tendrán los separadores de columnas. Debemos definir la anchura, la forma y el color de este, o bien podemos utilizar `column-rule-width`, `column-rule-style` y `column-rule-color` por separado para definirlo individualmente.

`column-span` se usa para determinar si elementos hijos siguen el formato de las columnas o no. Por ejemplo si tenemos un texto con un titulo gracias a esta propiedad podemos definir si el titulo se posiciona como el texto normal en la primera columna, o por el contrario utiliza todo el ancho de la parte superior de las columnas que haya definidas.

3.12. Propiedades de la caja (Box Properties)

A continuación veremos los nuevos controles que incorpora CSS3 para definir mejor los contenedores o cajas del contenido de nuestra web.

`overflow-x` se utiliza para definir el comportamiento de un elemento el cual desborda su caja contenedora por la derecha o por la izquierda.

`overflow-y` se utiliza para definir el comportamiento de un elemento el cual desborda su caja contenedora por arriba o por abajo.

`overflow-style` se utiliza para definir como se mostrara un elemento que ha desbordado su caja contenedora.

`rotation` se utiliza para hacer rotar una caja contenedora en los grados que nosotros le indiquemos. Para utilizar esta propiedad es necesario definir `rotation-point`.

`rotation-point` se utiliza para definir cual será el punto o eje sobre el cual rotara la caja contenedora. Se puede especificar mediante palabras claras o mediante una pareja de porcentajes haciendo referencia al eje de x y de y respectivamente.

`box-align` se utiliza para especificar el alineamiento que tendrán los elementos hijos de esa caja contenedora.

`box-direction` se utiliza para definir cual será la dirección en la que se mostraran los elementos hijos, se puede definir normal o que se muestren en orden inverso.

`box-flex` se utiliza para definir cual será el porcentaje de tamaño de esa caja contenedora respecto a las otras que estén dentro del mismo elemento padre en una relación a 1.

`box-flex-group` se utiliza para definir el porcentaje de tamaño que tendrá un grupo de cajas contenedoras, al igual que la propiedad anterior, esta se determina en relación a 1.

`box-lines` se utiliza para definir si todas las cajas contenedoras hijas serán mostradas en una sola fila o en múltiples filas. Solo podemos definir si se muestran en simple o en múltiples filas, será el propio navegador el que decida cuantas.

`box-ordinal-group` se utiliza para definir el orden en que se mostraran las cajas contenedoras. Con esta propiedad indicamos el número que ocuparan cuando sean mostrados.

`box-orient` se utiliza para determinar la orientación que tendrán las cajas contenedoras dentro de un mismo elemento padre. Pueden definirse en `horizontal`, `vertical`, `inline-axis`, `block-axis` o `inherit`.

`box-pack` se utiliza para definir cual será el emplazamiento que ocuparan las cajas contenedoras hijas cuando estas no sean de inferior tamaño que el elemento padre. Podemos definir que se posicionen en `start`, `end`, `center` o `justify`.

`grid-*` se utiliza para definir una cuadricula dentro de un elemento. Estas propiedades se utilizan para posicionar o escalar elementos hijos de manera más cómoda y sencilla. Podemos utilizar `grid-column` para definir las columnas y `grid-rows` para definir las filas.

3.13. Interfaz de usuario (User Interface)

A continuación encontraremos la descripción de aquellos elementos con los que el usuario interacciona cuando visualiza el documento web, cambiando la presentación de los diferentes elementos.

`appearance` se utiliza para definir el aspecto que tendrá un determinado elemento. Podemos definir su aspecto como `normal`, `icon`, `window`, `button`, `menu` o `field.`

`box-sizing` se utiliza para acotar el contenido de un elemento(caja). Puede definirse en `border-box` que acota el borde al contenido utilizando el espacio acotado, más el padding,

más el propio borde, tratándolo como si fuera borde; o buen `inherit` para heredar el valor de un elemento padre.

`icon` se utiliza apara utilizar un icono para estilizar, y por lo tanto representar, un determinado elemento de nuestro documento como un icono.

`resize` se utiliza para definir si un determinado elemento puede ser modificado en tamaño por el usuario. Esta propiedad nos permite definir si podemos modificarlo verticalmente, horizontalmente o en ambas direcciones, así como no permitirlo.

`nav-*` se utiliza para permitir al usuario navegar por la web con las flechas del teclado. Con estas propiedades podemos definir cual será el siguiente elemento al cual se dirija según que flecha haya pulsado. Con `nav-index` definimos el número identificador que tendrá ese elemento. Con `nav-up`, `nav-left`, `nav-right` y `nav-down` definimos hacia que elemento de los indicados en el índice nos dirigiremos una vez pulsemos la flecha arriba, izquierda, derecha o abajo respectivamente.

`outline-offset` se utiliza para determinar la distancia a la cual se mostrara el elemento `outline`. El elemento `outline` pertenece a CSS2, el cual se utiliza para marcar el contorno exterior de un elemento, por fuera del borde. Ahora con esta propiedad podemos emplearlo a la distancia que precisemos.

3.14. Valores de los colores (Color values)

Hasta ahora los colores en CSS podrían ser definidos de manera hexadecimal o mediante valores RGB, en css3 podemos definir los colores de una nueva manera aumentado la paleta.

Colores RGBA

Se usa de la misma manera que definiendo colores por RGB pero además contiene otro valor para indicar la transparencia en escala 0 – 1, donde 0 es transparente y 1 es opaco.

```
rgba(255,0,255,0.4);
```

Figura3_19: Ejemplo de código utilizando codificación de color RGBA.

Colores HSL

Mediante esta propiedad elegimos el color, la saturación y el brillo de este. Donde saturación y brillo son un porcentaje y el color es un número de 0 a 360 donde 0 es rojo, 120 verde y 240 azul.

```
hsl(230,40%,80%);
```

Figura3_20: Ejemplo de código utilizando codificación de color HSL.

Colores HSLA

Funciona exactamente igual que HSL pero con la diferencia que se ha añadido un valor para poder definir la transparencia. En escala 0-1 donde 0 es transparente y 1 es opaco.

```
hsla(230,40%,80%,0.6);
```

Figura3_21: Ejemplo de código utilizando codificación de color HSLA.

CAPITULO 4: CASO PRÁCTICO. PROPUESTA DE ACTUALIZACION DEL SITIO WEB: MPIu+u – Modelo de Proceso de la Ingeniería de la Usabilidad y de la Accesibilidad.

4.1. Contextualización del sitio MPIu+a

El sitio sobre el cual realizaremos una aplicación de todo lo visto anteriormente acerca de HTML5 y CSS3 será el sitio web de MPIu+a – Modelo de Proceso de la Ingeniería de la Usabilidad y de la Accesibilidad que se alberga en la siguiente dirección web: http://www.grihohcitools.udl.cat/mpiua/.

Esta web es una web con finalidad docente y gran parte del contenido proviene de la tesis doctoral del Dr. Toni Granollers i Saltiveri: *MPIu+a. Una metodología que integra la ingeniería del software, la interacción persona-ordenador y la accesibilidad en el contexto de equipos de desarrollo multidisciplinares*. El objetivo de este sitio web es dar soporte a docentes, alumnos y profesionales en todos los aspectos relacionados con el Diseño Centrado en el Usuario, la Usabilidad, La Accesibilidad y a la Interacción Persona-Ordenador en general.

Esta web es fruto de otros dos trabajos de final de carrera, por un lado *Implementación del sitio web www.mpiua.net* de Dunia Meler Pascual, y por otro lado *Aplicació de l'Enginyeria de la Usabilitat i de l'accessibilitat al desenvolupament de www.mpiua.net* de Marta Gonzalez Mata ambos realizados el año 2005.

4.2. Estudio de la utilización de HTML5 y CSS3 en el sitio web.

En este apartado vamos a ir punto por punto de los dos capítulos anteriores y vamos a ver cuales de ellos podemos aplicar nuestro sitio web.

Semántica de HTML5

Esta será la parte a tener más en cuenta para la actualización del sitio web. La pagina web entera tiene estructurado todo el contenido mediante la etiqueta div. Deberemos cambiar todas etiquetas por las nuevas que nos proporciona HTML5 para cada situación.

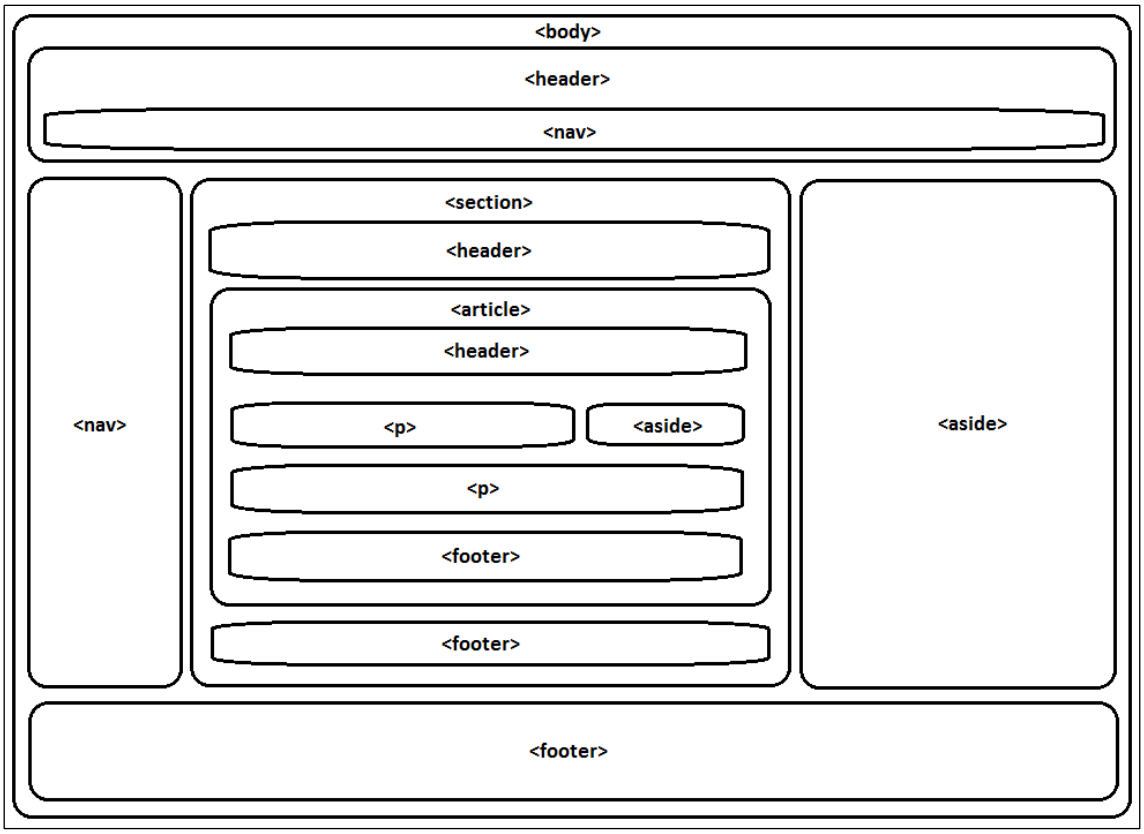

Figura4_01: Ejemplo tipo de utilización de etiquetas HTML5.

También tendremos que cambiar la declaración del doctype del documento, así como la declaración del charset y la agregación de ficheros externos como la hoja de estilos o archivos javascript.

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Titulo del documento</title>
        <link href="main.css" rel="stylesheet" >
        <script src="modernizr.js"></script>
    </head>
    <body>
        <p>Contenido</p>
    </body>
</html>
```

Figura4_02: Ejemplo de código en HTML5.

Formularios HTML5

En la web únicamente tenemos un formulario, que es el cuadro de búsqueda que hay en la esquina superior derecha. Pero no utilizaremos los nuevos formularios de HTML5 ya que utilizaremos un motor de búsqueda personalizado de Google, el cual nos proporciona su propio cuadro de búsqueda y no nos ofrece la posibilidad de editarlo.

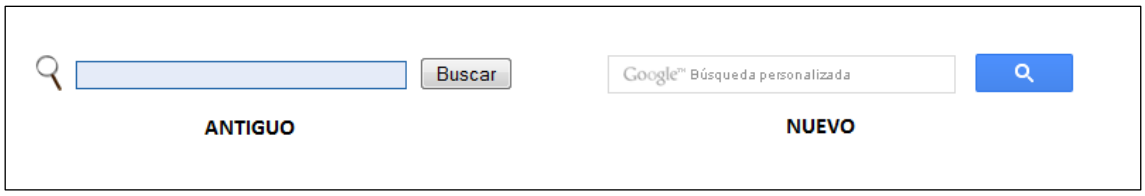

Figura4_03: Imagen con el antiguo y el nuevo buscador.

API de audio y video

Utilizando esta API podemos integrar en la propia web los videos que esta dispone. De este modo no será necesario redirigir al usuario hacia páginas externas ni tampoco la utilización de Flash, de modo que estará integrado con los demás contenidos de nuestra web.

Figura4_04: Imagen de un video incrustado en la nueva web, utilizando la API de video.

Resto de APIs

Del resto de APIs de HTML5 vistas anteriormente no utilizaremos ninguna ya que con la funcionalidad actual de la web no tienen cabida. La mayoría de las APIs a excepción de la de audio y video radican de la interacción del usuario con la web, y las otras están centradas en como procesar la información derivada de dicha interacción. Esta web es una web de divulgación por lo que la interacción de los usuarios no va mucho más allá de la propia consulta de los contenidos de esta.

CSS3

Dentro de nuestra web tienen cabida algunos de los nuevos aspectos que nos proporciona la nueva versión de CSS. Las animaciones, transformaciones y transiciones quedan fuera del área de aplicación ya que no lo amparamos en el contexto de la web.

En la hoja de estilos haremos uso de los nuevos selectores que nos brinda CSS3 para así no tener que marcar todos y cada uno de los elementos con un identificador único. De este modo solo tendremos que hacerlo en aquellos casos donde no podamos llegar con estos nuevos selectores.

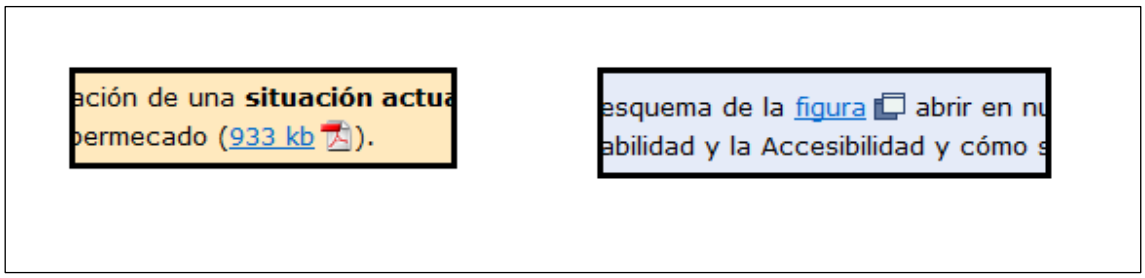

Figura4_05: Imagen de la utilización de selectores en la nueva web.

También utilizaremos el redondeado de esquinas para estilizar aquellos elementos con tal efecto.

Figura4_06: Imagen de la utilización de la propiedad de redondear esquinas.

4.3. Aplicación de los cambios.

A continuación veremos como hemos aplicado todas las características comentadas en el punto anterior, así como cualquier otro cambio que se haya realizado. La web se ha implementado nueva desde 0, para evitar en lo máximo posible, la utilización de antiguos elementos tanto de versiones anteriores de HTML como de CSS.

ANTIGUA WEB

NUEVA WEB

Figura4_07: Imagen de comparación entre ambas webs.

A pesar de ser una web totalmente nueva, se ha hecho un esfuerzo por mantener el diseño antiguo, ya que era el resultado de un estudio previo, el cual tenía buenos resultados y no era necesario modificarlo. Las modificaciones realizadas las explicaremos todas en los puntos venideros.

4.3.1 Utilización de HTML5.

En este punto veremos las novedades que contiene la nueva web en relación con HTML5.

Estructuración del documento utilizando la nueva semántica.

La implementación de la web, se ha basado en la utilización de las nuevas etiquetas de HTML5 para la estructuración del contenido, utilizando el siguiente esquema:

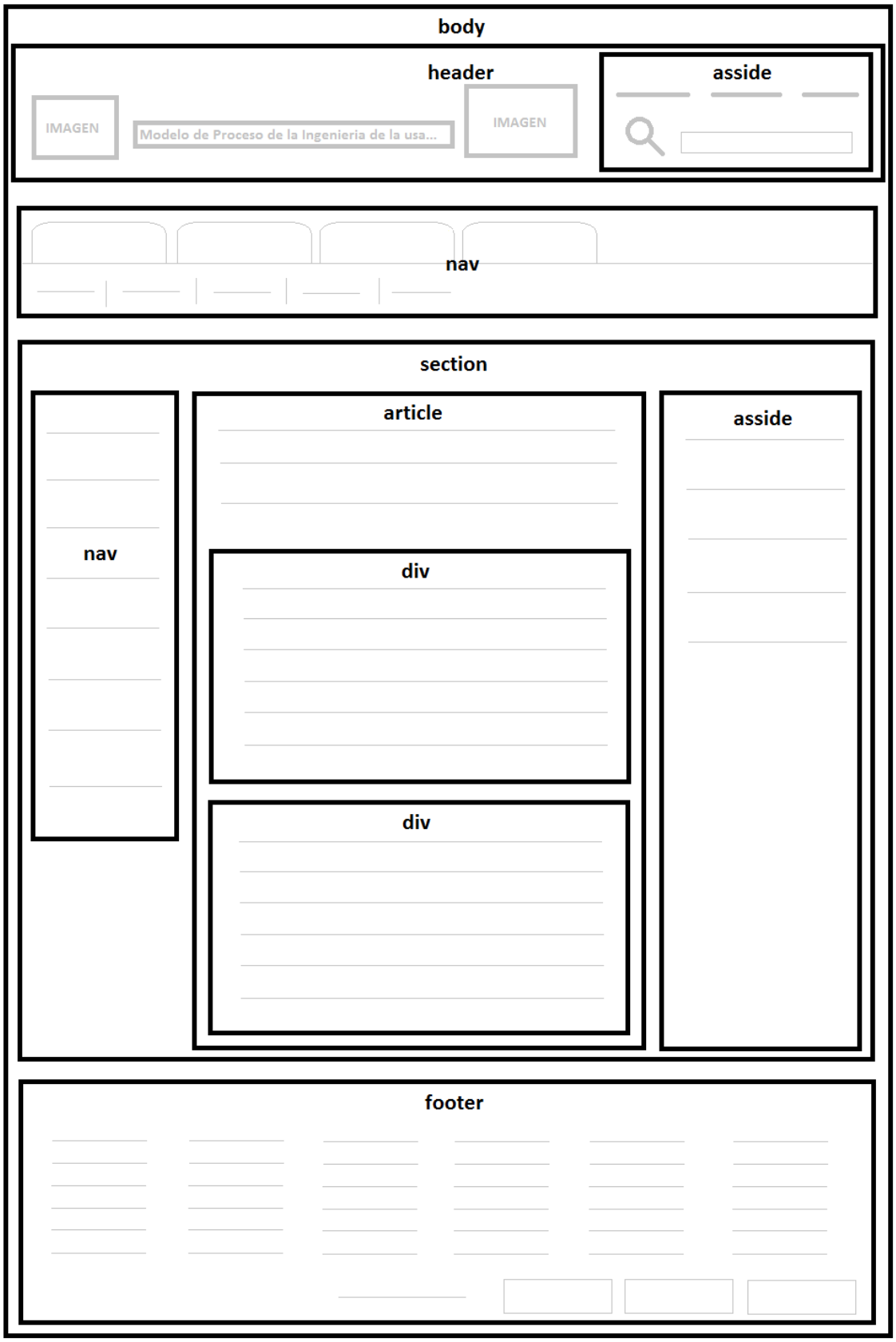

Figura4_08: Estructuración de la nueva web.

Como podemos apreciar en la figura4_02, se han utilizado las nuevas etiquetas para estructurar el contenido dentro de nuestra web.

Figura4_09: Cabecera de la nueva web.

En la parte superior hemos utilizado la etiqueta `header` que contiene la cabecera de la página. Este contiene el logo de MPIu+a, el titulo de la web, y un logo central del grupo GRIHO. En la zona derecha, se ha hecho uso de la etiqueta `aside` para englobar el contenido situado en esa posición, que son un conjunto de enlaces de carácter general y un cuadro de búsqueda.

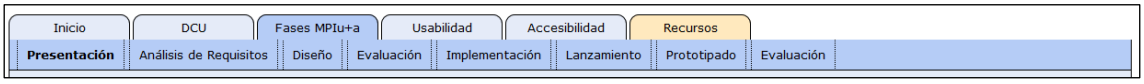

Figura4_10: Menú de navegación de la nueva web.

A continuación, se encuentra el menú de la web que podemos ver en la figura4_04, el cual esta dentro de la etiqueta `nav`. Esta etiqueta contiene el conjunto de enlaces que se utilizan para acceder a las distintas páginas del sitio web.

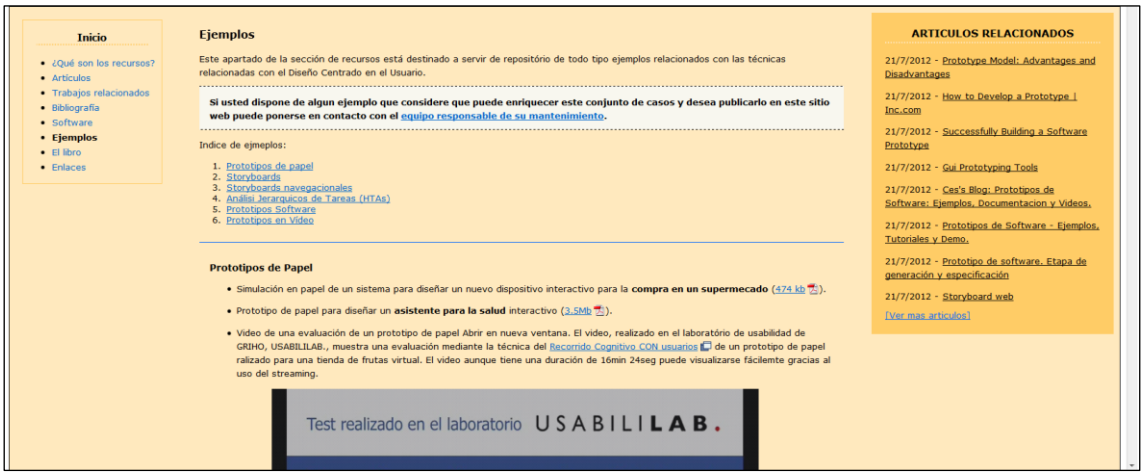

Figura4_11: Contenido de la nueva web. Es de diferente color ya que pertenece a la parte de recursos, y esa zona tiene este estilo.

Seguidamente, como vemos en la figura4_05 encontramos la etiqueta `section`, que es la que alberga el contenido de cada una de las paginas. Como en cada página solo tratamos un único tema, solamente utilizamos esta etiqueta una única vez. Dentro de esta etiqueta encontramos otra etiqueta `nav` que contiene otro bloque de enlaces, que direccionan a otras paginas de esta misma categoría o bien que guardan relación con la pagina actual. En la zona central se ha dispuesto la etiqueta `article` la cual alberga todo el contenido de esa pagina, que generalmente será texto. Dentro de esta etiqueta, en algunas paginas donde se divide el contenido en diferentes partes, se han implementado distintas etiquetas `div` para cada parte, para luego poder utilizar los puntos de ancla, así como el selector `:target` de CSS3 que nos permite estilizar esa parte, pero ya hablaremos de eso mas adelante. En la zona derecha se ha localizado la etiqueta `aside` que contiene artículos relacionados con la pagina actual, proporcionando un extra de contenido.

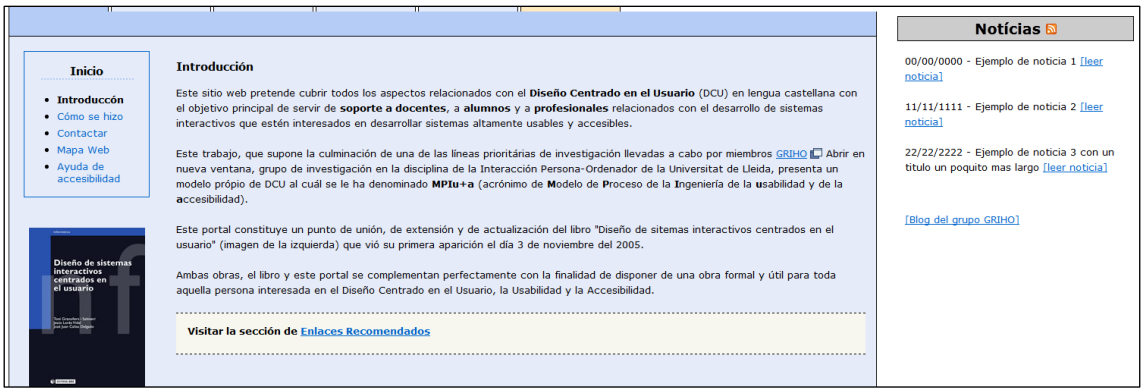

Figura4_12: Contenido de la nueva web en paginas de carácter general, en este caso perteneciente a la pagina principal.

Como vemos en la figura4_06, esta etiqueta en páginas de carácter general como la pagina principal, la de contacto, el mapa web,… no se ha implementado. En estos casos se ha implementado la etiqueta `aside` igualmente, pero no dentro de la etiqueta `section`, si no directamente en la etiqueta `body`. Ya que el contenido de esta no estará relacionado con cada una de estas paginas, si no que mostrara las entradas del blog del grupo GRIHO, imitando la implementación del antiguo sitio.

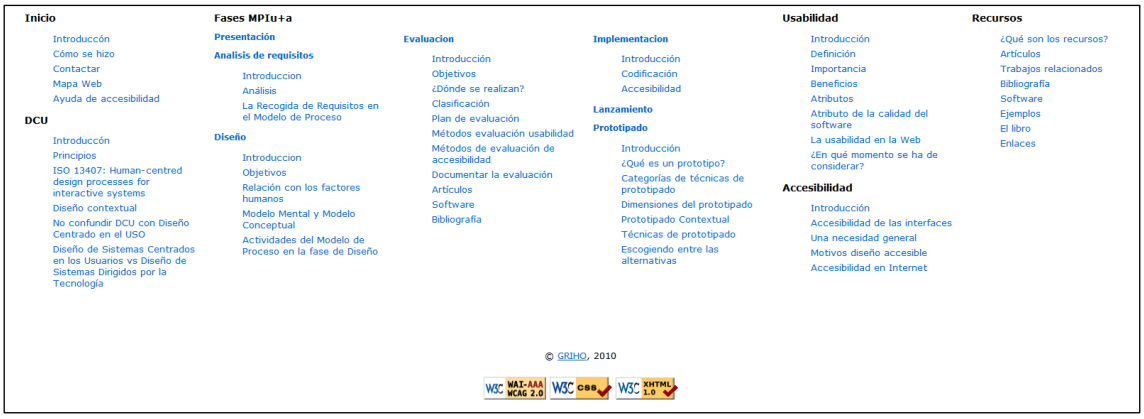

Figura4_13: Pie de pagina de la nueva web.

Para finalizar, como se puede apreciar en al figura4_07, en el pie de la pagina se ha posicionado la etiqueta `footer` que contiene el mapa web del sitio, y en su parte inferior los diferentes logos de conformidad de la W3C. En esta nueva página dichos logos se sitúan en la parte central.

Utilización de la nueva semántica en la definición del documento.

Para definir nuestro documento web hemos utilizado las siguientes líneas de código:

```
<!DOCTYPE HTML>
<html lang="es">
      <head>
      <meta charset="utf-8">
       <title>MPIu+a - Modelo de Proceso de la Ingeniería de la Usabilidad y
la accesibilidad</title>
       <link href="estil.css" rel="stylesheet">
       <!-- JQuery -->
       <script src="arxius/jq2.js"></script>
</head>
```

Figura4_14: Declaración del documento html.

Como podemos apreciar en la figura4_08, hemos utilizado la codificación adecuada para documentos HTML5. La correcta utilización del `DOCTYPE`, la utilización de la etiqueta `html` que como parámetro únicamente tiene el lenguaje; y la llamada a ficheros externos como la hoja de estilos o un archivo javascript, en este caso la librería JQuery.

La utilización de la API de video.

En la nueva web, mas concretamente en el apartado de recursos se ha implementado la utilización de la API de video de HTML5.

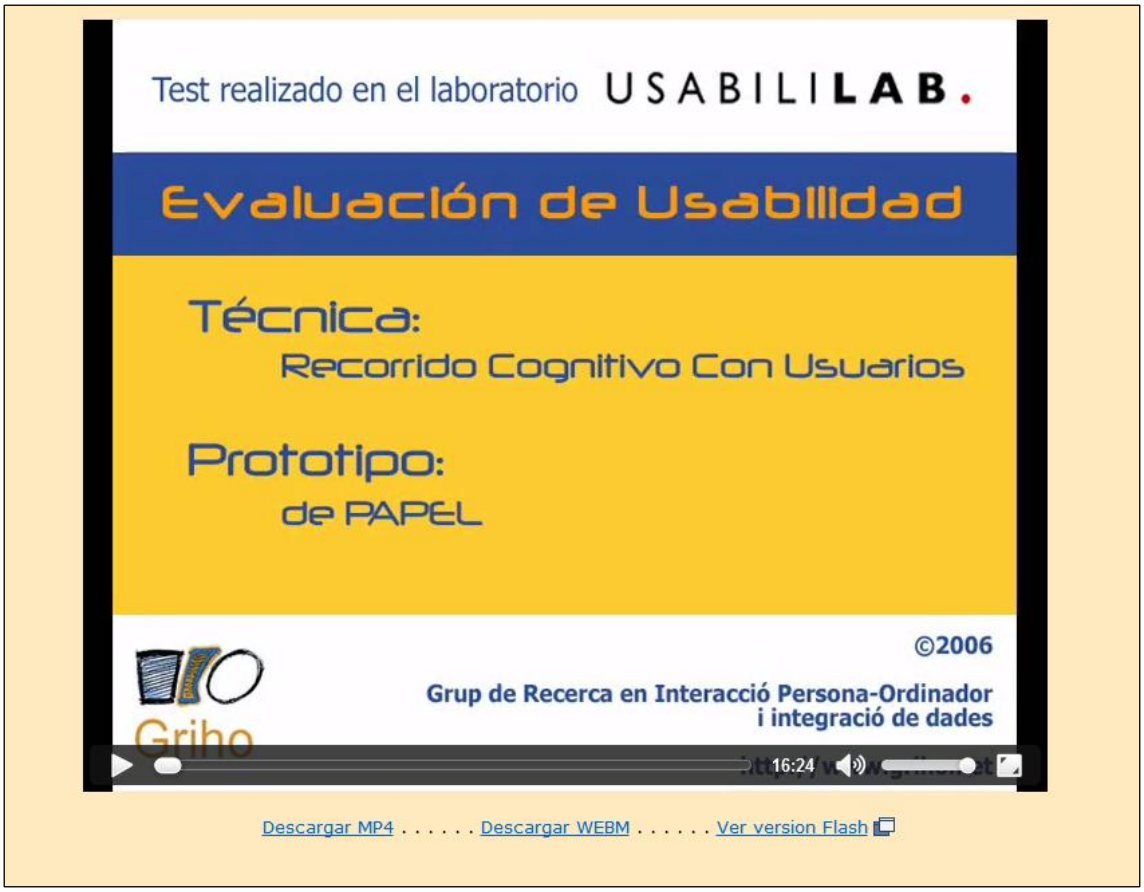

Figura4_15: Integración de video en la nueva web.

En la página de ejemplos, dentro del apartado de recursos se encontraban dos videos, pero en la anterior web se direccionaba a una página donde se mostraba el video mediante la

utilización de flash. En la nueva web, se ha incrustado el video en la propia página gracias a la nueva API de HTML5. El video se ha codificado en MP4 y WEBM donde cada navegador utiliza el que soporta. Con estos dos formatos se da soporte a todos los navegadores que actualmente hay en el mercado. Seguidamente del video, se pone a disposición del usuario la descarga directa del video en ambos formatos, así como un link que proporciona acceso a la antigua página con la versión flash del video.

Implementación del mapa web.

En la nueva web se ha implementado dentro de la etiqueta `footer` el mapa de la web. Este mapa se muestra en todas las páginas.

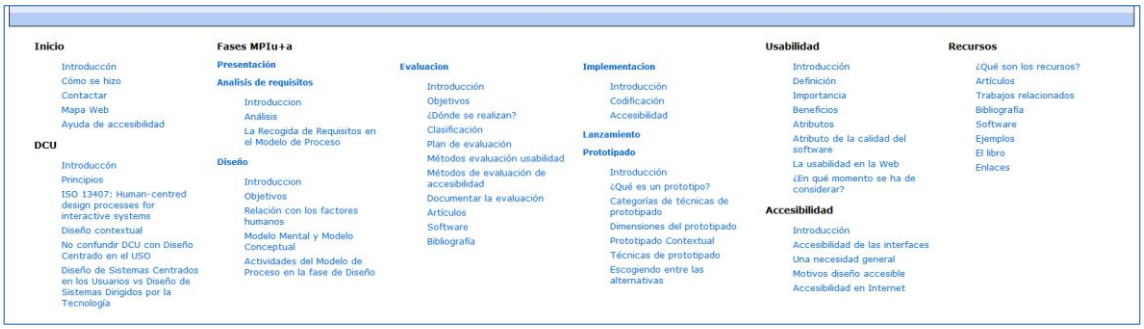

Figura4_16: Mapa web implementado en la nueva web.

Como podemos ver en la figura4_10 el mapa web contempla las distintas páginas que forman el sitio web. Se han organizado por las diferentes etiquetas del menú, que corresponden a los distintos apartados del sitio. Cada titulo del cada bloque está diferenciado del resto por tener un tamaño superior y por tener el texto de color negro. Dentro del apartado de las fases de MPIu+a hay otro estilo de texto que son mas grandes y en negrita, estos enlaces pertenecen a los sub apartados pertenecientes a las distintas fases.

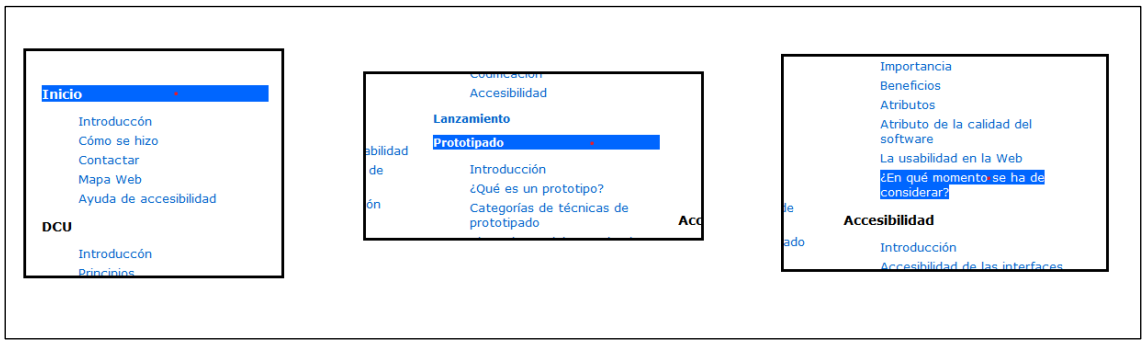

Figura4_17: Muestra de la acción de los enlaces del mapa web al posicionar el puntero sobre estos. El punto rojo de las imágenes significa la posición del puntero.

Como podemos apreciar en la figura4_11, siguiendo el estilo del sitio, cuando pasamos el puntero sobre los enlaces todos ellos tienen un fondo azul y las letras blancas.

4.3.2. Utilización de CSS3.

En este punto veremos las novedades que contiene la nueva web en relación con CSS3.

Utilización de la propiedad `border-radius`.

Con esta propiedad conseguimos el efecto de redondeado de las esquinas de los menús.

Figura4_18: Utilización de la propiedad para redondear las esquinas.

Para la implementación de estos menús hemos utilizado el siguiente código:

```
border-radius:10px 10px 0px 0px;
```

Figura4_19: Código para redondear las esquinas de los menús.

De la figura4_13 podemos interpretar que los dos primeros parámetros pertenecen a las esquinas superiores, empezando por la izquierda. Las otras dos esquinas se han mantenido a cero ya que no nos interesa redondearlas para crear así el aspecto de pestaña.

Utilización de los selectores `[atributo$=valor]` y `[atributo^=valor]`.

En la página se han utilizado estos selectores para implementar las imágenes o iconos que aparecían en la antigua web al lado de los enlaces a páginas externas o a enlaces de archivos pdf. Estas imágenes eran insertadas manualmente junto con el resto del código de la pagina, ahora gracias a CSS3 podemos hacerlo de manera automática y de una manera muy sencilla.

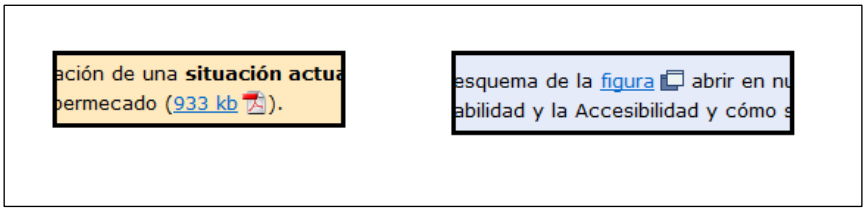

Figura4_20: Imagen de muestra de iconos que aparecen junto a los enlaces.

Para lograr este cometido hemos utilizado el siguiente código:

```
section article a[href$=".pdf"]{
	padding-right:20px;
	background:url(../imatges/logopdf.gif) no-repeat right center;
}

section article a[href^="http://"]{
	padding-right:20px;
	background:url(../imatges/newwind.gif) no-repeat right center;
}
```

Figura4_21: Código de la utilización de selectores según el contenido de los atributos.

Gracias a estos selectores en todos los enlaces de dentro del articulo aparecerán, en el primer caso, el logo de un archivo pdf al lado de un enlace cuya terminación sea la cadena ".pdf". En

el segundo caso, se mostrara el icono de nueva ventana cuando el enlace empiece por la cadena “http://”.

Utilización del selector `:target`.

En la nueva web, en aquellas páginas donde hay puntos de ancla a distintos puntos de la misma pagina, se ha implementado una remarcación del destino a los cuales va dirigido tal enlace.

Figura4_22: Utilización del selector :target en la nueva web.

Para realizar este efecto tan resultón sencillamente es necesario el siguiente código:

```
section article div:target {
      border:2px dashed #B5CCF6;
}
```

Figura4_23: Código de la utilización del selector :target.

Este selector funciona por la modificación en la URL ya que se añade el identificador del destino al que hace referencia el enlace del ancla. Para quitar este efecto basta con utilizar otra ancla, por ejemplo, utilizando el enlace de “Inicio de ventana”.

La razón de las etiquetas `div` dentro de la etiqueta `article` que hemos visto en el punto *4.3.1 Utilización de HTML5*, es para poder utilizar tal efecto y englobar todo el párrafo o conjunto de párrafos, enlaces y demás contenido bajo la misma etiqueta, pues el borde que dibujamos se traza alrededor de esta etiqueta `div`.

4.3.3. Utilización de otras tecnologías.

En este punto vamos a ver todos aquellos elementos que utilizan otras tecnologías que no son ni HTML5 ni CSS3.

El buscador de Google.

El cuadro de búsqueda posicionado en la esquina superior derecha, esta implementado por el motor de búsqueda personalizado para sitios web de Google.

Figura4_24: Cuadro de búsqueda proporcionado por Google.

Para obtener este cuadro de búsqueda, hemos accedido en la dirección http://www.google.es/cse/ a través de nuestra cuenta de Google y hemos creado nuestro motor de búsqueda personalizado. En este caso se ha creado un motor que busca los resultados en el dominio de la web que en este caso es: http://www.grihohcitools.udl.cat/mpiua/.

Seguidamente se nos ha proporcionado un código de la figura4_14 en lenguaje javascript para incrustar en nuestra web.

```
<script>
          (function() {
            var cx = '014670605655477939748:y6ba8mtztby';
            var gcse = document.createElement('script'); gcse.type =
'text/javascript'; gcse.async = true;
            gcse.src = (document.location.protocol == 'https:' ? 'https:' :
'http:') +
                '//www.google.com/cse/cse.js?cx=' + cx;
            var s = document.getElementsByTagName('script')[0];
s.parentNode.insertBefore(gcse, s);
          })();
</script>
```

Figura4_25: Código de búsqueda proporcionado por Google.

Si se desea cambiar el motor de búsqueda para cambiar el domino o la cuenta o cualquier otro cambio basta con cambiar el parámetro de la variable `cx`, ya que es la que contiene toda la información acerca de nuestro motor personalizado. También se nos ha proporcionado el código de la figura4_15, que debemos insertar en el emplazamiento donde queremos que aparezca el cuadro de búsqueda.

```
<gcse:searchbox-only></gcse:searchbox-only>
```

Figura4_26: Código del cuadro de búsqueda proporcionado por Google.

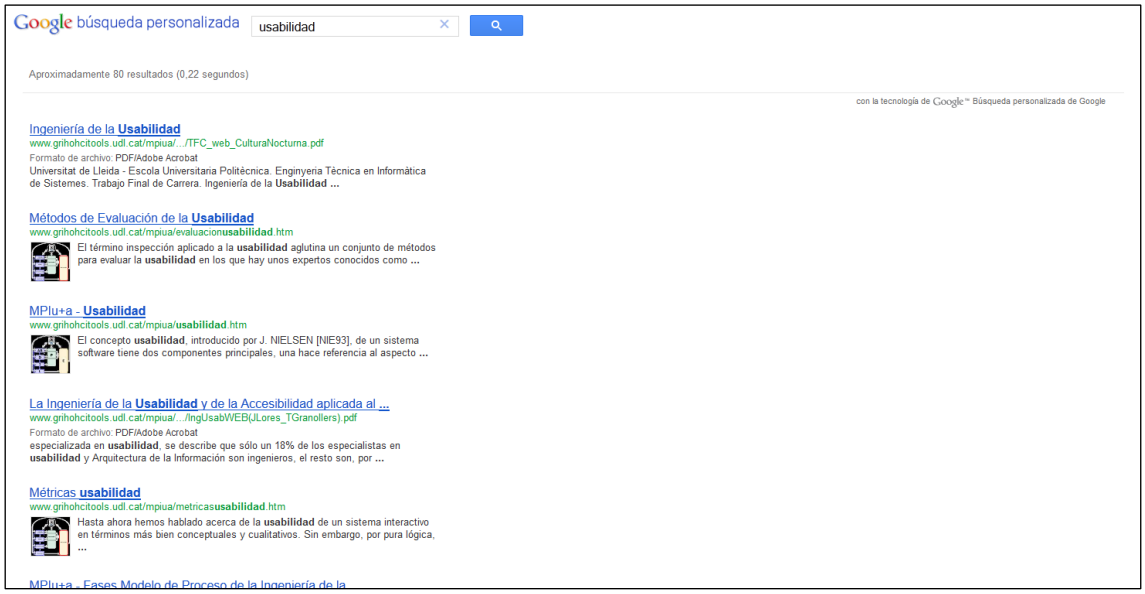

Figura4_27: Imagen del resultado de una búsqueda utilizando el motor de búsqueda personalizado de Google.

Como podemos ver en la figura4_16, este es el resultado de una búsqueda, en este caso la palabra “usabilidad”. El cuadro de búsqueda nos direcciona a esta página que únicamente contiene lo que se aprecia en la imagen.

Este método para crear motores de búsqueda, esta muy bien por la sencillez de su implementación, ya que únicamente debemos entrar en la pagina de google, introducir el dominio sobre el cual queremos realizar la búsqueda y posteriormente ubicar en nuestra web el código proporcionado. Por el contrario, la gran desventaja es la personalización del cuadro de búsqueda así como la obtención de resultados, que es totalmente nula. Únicamente se nos da la opción de elegir entre una muy limitada gama de colores, algo muy insuficiente si queremos controlar la accesibilidad de nuestra web, pues de sobras es conocido el desinterés en este aspecto por parte de esta compañía.

El nuevo menú.

En esta web se ha implementado un nuevo menú de una manera diferente, pero manteniendo el mismo diseño en la medida de lo posible.

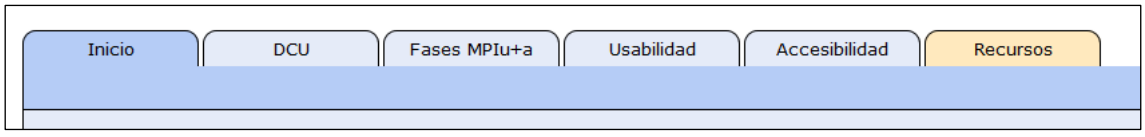

Figura4_28: Imagen del menú de la nueva web.

La principal diferencia dista, en que todos los efectos de este menú se han realizado en la hoja de estilos. La más notoria la propiedad de redondear las esquinas gracias a la nueva propiedad de CSS3 vista anteriormente.

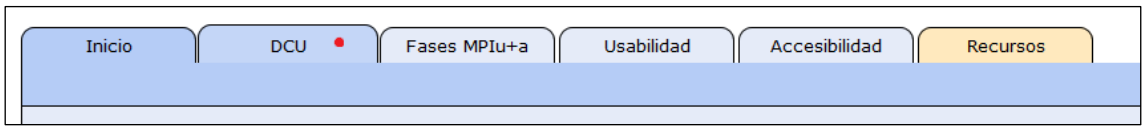

Figura4_29: Imagen de la utilización del menú de la nueva web.

El punto rojo de la figura4_18 simboliza la posición del puntero, que en este caso puede apreciarse como se magnifica ligeramente la pestaña y cambia el color de fondo quedándose entremedio de una pestaña actualmente seleccionada y una sin seleccionar. Para la pestaña recursos se utiliza un color naranja más oscuro para obtener el mismo efecto.

Figura4_30: Imagen de la utilización del menú de la nueva web.

En la figura4_19 se aprecia que cuando desplazamos el puntero por encima de la pestaña de Fases MPIu+a aparece el submenú de este apartado. Para que se muestre este submenú, se utiliza javascript que captura los eventos de cuando desplazamos o quitamos el puntero de esta pestaña. Del mismo modo se ha realizado en el propio submenú, para que este pueda seguir mostrándose. Además se ha proporcionado un retardo de un segundo en la desaparición de este al apartar el puntero del ratón. Esto se ha hecho por si nos apartamos momentáneamente del submenú mientras lo recorremos, para que este no desaparezca al momento, y no nos obligue a volver a la pestaña de Fases de MPIu+a para volver a hacer aparecer de nuevo este submenú.

Figura4_31: Imagen de la utilización del menú de la nueva web.

En la figura4_20 podemos apreciar que al pasar el ratón por los distintos elementos del submenú, este cambia el fondo a un azul mas vivo y cambia el color del texto al color blanco, para crear un mejor contraste.

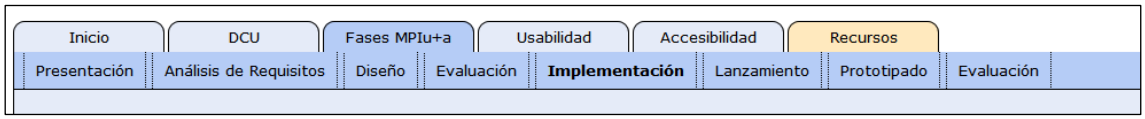

Figura4_32: Imagen de la utilización del menú de la nueva web.

Una vez hemos clicado en el link, y una vez hemos sido redirigidos a esa página, el aspecto del submenú no difiere demasiado al estilo de la web original.

La cabecera emergente.

En esta nueva web se ha implementado una nueva cabecera que aparece cuando nos desplazamos hacia abajo en nuestra web.

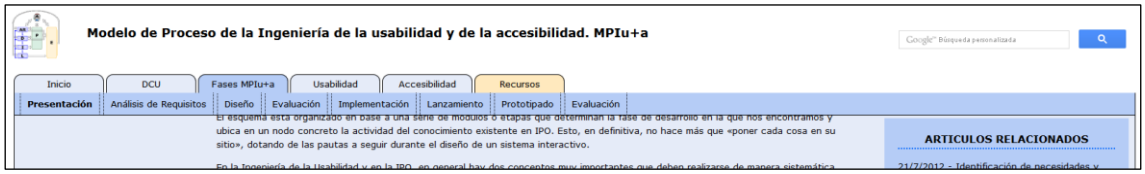

Figura4_33: Imagen de la cabecera emergente.

Esta segunda cabecera, que puede verse en la figura4_22, contiene el logo de la web en un tamaño reducido, así como el titulo de la misma reducido de tamaño también. En la parte derecha se ha posicionado el cuadro de búsqueda proporcionado por Google. También se ha dispuesto de un menú, el cual tiene la misma implementación que el explicado en el apartado anterior. Esta cabecera se posiciona fijamente en la parte superior de la pantalla y se muestra siempre que no estemos arriba del todo de la página, pues entonces se visualiza la cabecera del sitio.

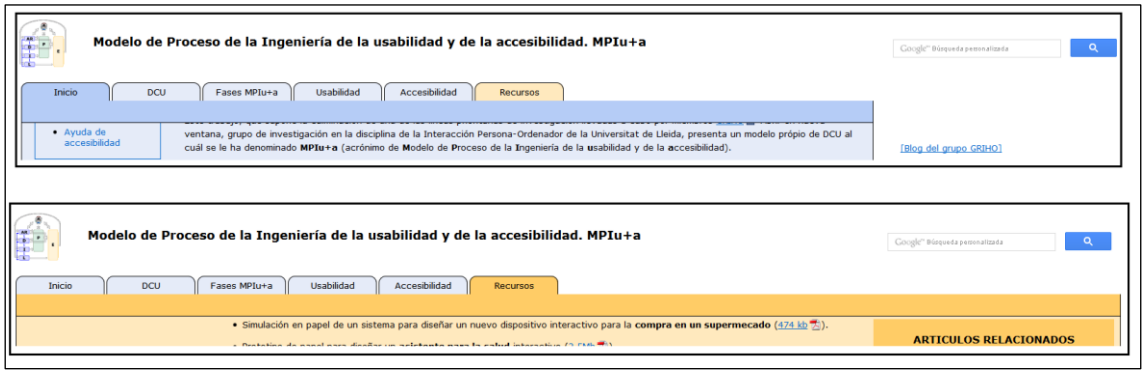

Figura4_34: Imagen de las distintas cabeceras de la nueva web.

En la parte de recursos el color del menú cambia al naranja propio de esa página, y en las páginas de carácter general se ha reducido su tamaño para cuadrar con el recuadro del contenido.

Para implementar esta cabecera utilizamos javascript y el evento window.scroll para conocer en todo momento la posición de la ventana y mostrar o no esta segunda cabecera.

Implementación del cuadro de artículos relacionados.

En la nueva web, se ha dispuesto de un cuadro a la derecha de la página con contenido para enriquecer la propia página web. Este contenido extra se aporta utilizando una herramienta externa de bookmarking, en este caso Zootool (http://zootool.com).

Una vez registrados en esta pagina, se nos proporciona el elemento que vemos en la figura4_24 que debemos insertar en la barra de marcadores de nuestro navegador.

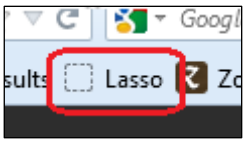

Figura4_35: Imagen de la barra de marcadores.

Cuando navegando por la web, encontramos un recurso que queramos agregar, basta con pulsar este marcador y se nos re direccionará a la pagina que podemos observar en la figura2_25.

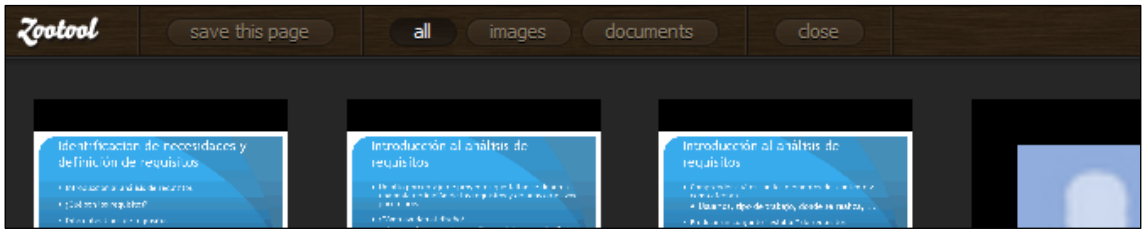

Figura4_36: Imagen de la parte superior al agregar un enlace en zootool.

Esta herramienta nos permite almacenar webs completas, imágenes, videos o documentos. Estas diferentes opciones se nos aparecen en la parte superior. La pagina que queremos guardar en la figura2_25, es una presentación de diapositivas y únicamente nos da la posibilidad de guardar las distintas paginas como imágenes o bien el documento entero, de ahí que no aparezca la opción de video. Si pulsamos el botón “save this page” almacenaremos la web completa y si presionamos el botón close cancelaremos la acción de guardado.

Una vez elegida nuestra opción de guardado se mostrara la imagen de la figura2_26.

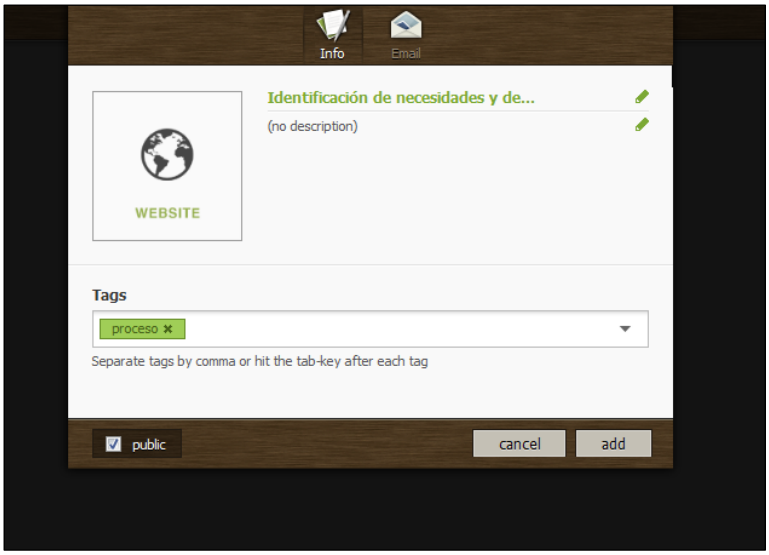

Figura4_37: Imagen del almacenado de una pagina en zootool.

Una vez llegado aquí, elegimos el titulo de esta entrada, que será el que se muestre en nuestra web, y mas abajo introducimos las etiquetas para marcar este elemento. Una vez pulsado el botón add ya se ha almacenado en nuestra cuenta y ya esta disponible en nuestra web. La descripción del enlace no es necesario para el correcto funcionamiento del sitio web MPIu+a ya que no se mostrara, por lo tanto no es imprescindible.

Figura4_38: Imagen del cuadro de los artículos de la nueva web.

Como vemos en la figura2_27 se ha añadido en primera posición el elemento que acabamos de añadir, puesto que están ordenados por fecha. Para cada sección de cada página se ha procurado una etiqueta para cada una, en este caso la etiqueta proceso es para la sección de Fases MPIu+a, para la sección de DCU utilizamos la etiqueta diseño y para las secciones de usabilidad, accesibilidad y recursos utilizamos las etiquetas con los mismos nombres respectivamente.

Estos datos se muestran utilizando de la API de zootool, que accedemos desde nuestro documento empleando la librería jQuery. La disposición de esta información puede ser modificada fácilmente, pues únicamente es necesario modificar el parámetro tag de la cadena de conexión a la API para que se muestren los resultados que nosotros queramos.

Figura4_39: Imagen del cuadro de los artículos de la nueva web.

La figura4_28 pertenece a la página de recursos, de ahí su cambio de color. Este recuadro de artículos únicamente contiene 8 enlaces, en la parte inferior se proporciona un enlace directo con la página personal de zootool donde se muestran los demás recursos con la misma etiqueta.

Gracias a la ayuda de jQuery, este cuadro se ha hecho flotante, y se desplaza con nosotros a medida que nos desplazamos por la web mostrándose siempre visible; siempre y cuando no nos salgamos del marco del elemento `section`, pues este es el emplazamiento que ocupa la etiqueta `aside` que lo contiene.

Flecha emergente.

En esta nueva web se ha implementado una flecha que aparece en la esquina inferior derecha, que al pulsarla nos transporta al inicio de la pagina.

Figura4_40: Imagen de la flecha emergente en la nueva web.

La figura4_29 es la imagen que se muestra en la web cuando nos desplazamos hacia abajo. Al posicionar el puntero encima cambia su aspecto como se muestra en la figura4_30.

Figura4_41: Imagen de la flecha emergente en la nueva web.

Si finalmente pulsamos iremos al inicio de la página. Para hacer aparecer o desaparecer esta flecha, utilizamos la librería jQuery y el evento window.scroll para hacerla aparecer y desaparecer.

4.3.4. Validación de los documentos.

Para ofrecer una correcta experiencia de usuario y deshacernos de cualquier error, en nuestra web, se han pasado los teses de validación de HTML5 y CSS3 que pone a nuestra disposición la W3C.

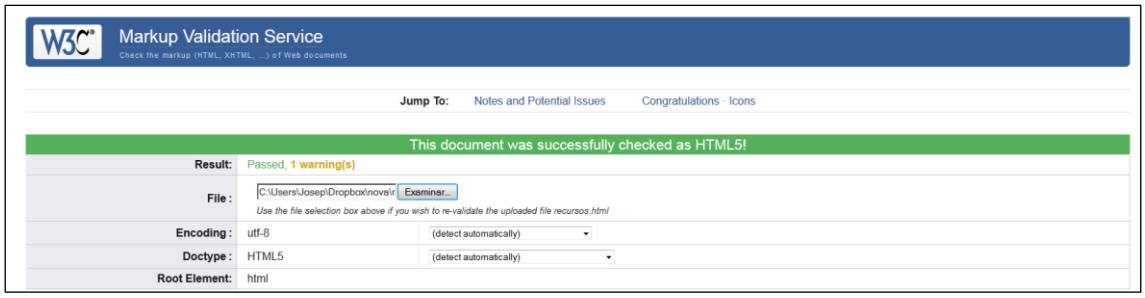

Figura4_42: Imagen que muestra la superación del test de HTML5.

Figura4_43: Imagen que muestra la superación del test de CSS3.

Un documento que contiene errores en su implementación, puede variar su aspecto dependiendo de cada navegador. Por eso de esta forma ya nos evitamos este contratiempo, pues con HTML5 y CSS3 ya nos encontramos con este problema en muchos casos.

Otro aspecto también muy importante es proporcionar la máxima accesibilidad al sitio web, quitar todas las barreras posibles, para que nuestro sitio pueda ser utilizado por el mayor número de usuarios posible. Para ello también se ha realizado el test de triple A de conformidad, eliminando los errores automáticos de todas las paginas que componen el sitio.

Figura4_44: Imagen que muestra que no hay errores automáticos en el test de accesibilidad.

Por lo tanto, una vez pasados estos teses, podemos mostrar en nuestra página los distintos sellos que acreditan la validación del sitio en estos distintos teses, como podemos apreciar en la figura4_34.

Figura4_45: Imagen que muestra los sellos que aparecen en la nueva web.

La validación de la accesibilidad no es un objetivo en este trabajo, pero tiene mucho sentido en el contexto del contenido del sitio web, así como, del grupo de investigación en el cual esta desarrollada. Por ese motivo se han revisado los errores automáticos, pero seria necesario hacer una verificación de todos los errores manuales y algún test de usuario, objetivos que escapan del alcance de este trabajo.

CAPITULO5: CONCLUSIONES Y TRABAJO FUTURO.

La realización de este trabajo me ha servido para conocer bastante más de cerca las nuevas tecnologías como son HTML5 y CSS3. Y me he dado cuenta que son mucho más potentes de lo que a priori imaginaba; a medida que me sumergía veía que su utilización va mucho más lejos que la simple implementación de paginas web.

Estoy muy satisfecho, aunque no por dominar, pero si por conocer la magnitud de estos venideros estándares, que creo firmemente que estarán muy presentes en nuestro día a día.

Además algo que quería realizar desde hacia mucho tiempo, era implementar un portal web accesible y usable, y gracias a este trabajo se me ha brindado la oportunidad de realizar una propuesta de actualización, utilizando las tecnologías estudiadas, del portal www.mpiua.net. Además contando con la ayuda inestimable del Dr. Toni Granollers, que es el director de este trabajo que me ha ayudado a asentar mejor todavía los conceptos aprendidos en la titulación, acerca de usabilidad y accesibilidad referentes a paginas web.

Como trabajo futuro y posterior al realizado en este trabajo, hay una infinidad de posibilidades. Cualquier tipo de aplicación web que utilizara alguna de las APIs de HTML5, puesto que en este trabajo únicamente se ha utilizado la de video.

Una de estas posibilidades seria la de hacer que el contenido el portal web, sea todavía más dinámico que el implementado en este trabajo, que consiste en una herramienta de bookmarking. La opción seria de hacerlo de alguna otra forma mucho más apetitosa e interesante para los usuarios.

Otra posibilidad seria un espacio donde usuarios de la página pudieran publicar ejemplos de prototipado, con una previa aprobación, en el portal web.

Otra posibilidad podría ser la de hacer que el contenido dinámico añadido sea social, donde los profesionales puedan emitir sus opiniones acerca de tal contenido añadido, para que los estudiantes puedan aprender de las valoraciones y razonamientos; el entender porque una cosa es buena o no y el adquirir una buena capacidad de critica, creo que podría ser fortalecida con este tipo de contenido.

CAPITULO 6: BIBLIOGRAFIA

LIBROS CONSULTADOS:

[1] WAY J. (2012) *Decoding HTML5*. Rockable

[2] FREEMAN E., ROBSON E. (2011) *Head First HTML5 Programming*. 1ª edición. USA: O'Relly. ISBN 978-1-449-39054-9.

[3] HOGAN P. (2010) *HTML5 and CSS3. Develop with Tomorrow's Standards Today*. USA: Pragmatic Programmers. ISBN 978-1-934356-68-5.

[4] GOLDSTEIN A.; LAZARIS L.; WEYL E. (2011) *HTML5 & CSS3 for the real world*. 1ª edición. Australia: Sitepoint. ISBN 978-0-9808469-0-4.

[5] BOWERS H.; SYNODINOS D.; SUMMER V. (2011) *Pro HTML5 and CSS3 Desing Patterns*. USA: Springer Science+Busines Media. ISBN 978-1-4302-3781-5.

[6] LUBBERS P.; ALBERS B.; SALIM F. (2011) *Pro HTML5 Programming*. USA: Springer Science+Busines Media .2ª edición. ISBN 978-1-4302-3865-2.

[7] MELER PASCUAL D. (2005) *Implementación del sitio web www.mpiua.net.* Escola Politècnica Superior. Universitat de Lleida.

[8] GONZALEZ MATA M. (2005?) *Aplicació de l'Enginyeria de la Usabilitat i de l'accessibilitat al desenvolupament de www.mpiua.net*. Escola Politècnica Superior. Universitat de Lleida.

RECURSOS DE INTERNET:

HTML5 Tutorial; http://www.w3schools.com/html5/default.asp [Consultado: Agosto 2012]

CSS3 Tutorial; http://www.w3schools.com/css3/default.asp [Consultado: Julio 2012]

World Wide Web Consorcium (W3C); http://www.w3.org [Consultado: Agosto 2012]

Making Form Fabulous with HTML5 – HTML5 Rocks; http://www.html5rocks.com/en/tutorials/forms/html5forms/ [Consultado: Julio 2012]

HTML5 Forms input types, attributes and new elements – demos, tips and tricks – Robert's Talk; http://robertnyman.com/2011/08/16/html5-forms-input-types-attributes-and-new-elements-demos-tips-and-tricks/ [Consultado: Julio 2012]

W3C HTML5 Tutorial – HTML5 TUTORIAL; http://www.w3ctutorial.com/html5-tutorial/html5-home [Consultado: Julio 2012]

Understanding HTML5 Semantics – Part 1: New elements | Adobe Developer Connection; http://www.adobe.com/devnet/dreamweaver/articles/understanding-html5-semantics.html [Consultado: Julio 2012]

Understanding HTML5 Semantics – Part 2: Document structure and global attributes | Adobe Developer Connection; http://www.adobe.com/devnet/dreamweaver/articles/understanding-html5-semantics-pt2.html [Consultado: Julio 2012]

Understanding HTML5 Semantics – Part 3: Changed and abasent elements | Adobe Developer Connection; http://www.adobe.com/devnet/dreamweaver/articles/understanding-html5-semantics-pt3.html [Consultado: Julio 2012]

HTML5 – Edition for web developers; http://developers.whatwg.org/ [Consultado: Julio 2012]

Article Archive | HTML5 Doctor; http://html5doctor.com/article-archive/ [Consultado: Julio 2012]

Semantics – HTML5 Rocks; http://www.html5rocks.com/en/features/semantics [Consultado: Julio 2012]

HTML5 – Wikipedia, la enciclopedia libre; http://es.wikipedia.org/wiki/HTML5 [Consultado: Agosto 2012]

CSS3 Previews – CSS3 .Info; http://www.css3.info/preview/ [Consultado: Julio 2012]

El API History de HTML5 en IE10; http://www.desarrolloweb.com/articulos/api-history-html5.html [Consultado: Agosto 2012]

Manipulating the Browser History | Mozilla Developer Network; https://developer.mozilla.org/en-US/docs/DOM/Manipulating_the_browser_history [Consultado: Agosto 2012]

Window.onpopstate |Mozilla Developer Network; https://developer.mozilla.org/en-US/docs/DOM/window.onpopstate [Consultado: Agosto 2012]

Window.history | Mozilla Developer Network; https://developer.mozilla.org/en-US/docs/DOM/window.history [Consultado: Agosto 2012]

HTML5 – Drag and Drop ; http://www.tutorialspoint.com/html5/html5_drag_drop.htm [Consultado: Agosto 2012]

8.7 Drag and Drop – HTML Standard; http://www.whatwg.org/specs/web-apps/current-work/multipage/dnd.html [Consultado: Agosto 2012]

Drag and Drop | Mozilla Developer Network; https://developer.mozilla.org/en-US/docs/DragDrop/Drag_and_Drop [Consultado: Agosto 2012]

Drag Operations | Mozilla Developer Network; https://developer.mozilla.org/en-US/docs/DragDrop/Drag_Operations#dragdata [Consultado: Agosto 2012]

HTML5: Web Sockets | inmensia; http://www.inmensia.com/blog/20110125/html5_web_sockets.html [Consultado: Agosto 2012]

Introducing Web Socket: Bringing Sockets to the web – HTML5 Rocks; http://www.html5rocks.com/es/tutorials/websockets/basics/ [Consultado: Agosto 2012]

WebSocket en HTML5 – Rodrigo Asensio; http://www.rodrigoasensio.com/2010/11/websockets-en-html5/ [Consultado: Agosto 2012]

HTML5 WebSockets Tutorial; http://www.tutorialspoint.com/html5/html5_websocket.htm [Consultado: Agosto 2012]

Almacenamiento con HTML5: Almacenamiento sql – Theproc.es; http://theproc.es/2011/4/21/18450/almacenamiento-con-html5--almacenamiento-sql [Consultado: Agosto 2012]

HTML5 Web SLQ Database; http://www.tutorialspoint.com/html5/html5_web_sql.htm [Consultado: Agosto 2012]

Tutorial: Web SQL Database; http://marakana.com/s/tutorial_web_sql_database,208/index.html [Consultado: Agosto 2012]

W3C: HTML5 Spec Due in 2014 – Application Developement – News & Reviews – eWeek.com; http://www.eweek.com/c/a/Application-Development/W3C-HTML5-Spec-Due-in-2014-108529/ [Consultado: Agosto 2012]

HTML5 will be done in 2014, What Comes Next? | Webmonkey | Wired.com; http://www.webmonkey.com/2011/02/html5-will-be-done-in-2014-what-comes-next/ [Consultado: Agosot 2012]

VIDEO-TUTORIALES:

WEINMAN B. (2011) LYNDA.COM | HTML5 Geolocation in Depth

WEINMAN B. (2011) LYNDA.COM | HTML5 Managing Browser History

WEINMAN B. (2012) LYNDA.COM | HTML5 Background Process with Web Workers

MARINI J. (2011) LYNDA.COM | HTML5 File API in Depth

WILLIAMSON J. (2010) LYNDA.COM | CSS3 First Look

WEINMAN B. (2011) LYNDA.COM | HTML5 Drag and Drop in Depth

WEINMAN B. (2012) LYNDA.COM | HTML5 Document Editing in Depth

MARINI J. (2011) LYNDA.COM | HTML5 Graphics and Animation with Canvas

WEINMAN B. (2011) LYNDA.COM | HTML5 Local Storage and Offline Applications in Depth

WEINMAN B. (2012) LYNDA.COM | HTML5 Messaging and Communications in Depth

WILLIAMSON J. (2011) LYNDA.COM | HTML5 Structure, Syntax and Semantics

HEFFERMAN S. (2011) LYNDA.COM | HTML5 Video and Audio in Depth

WILLIAMSON J. (2010) LYNDA.COM | HTML5 Video Tutorials

MARINI J. (2011) LYNDA.COM | HTML5 Web Forms in Depth